The development of a novel method to identify and describe driving events using only MEMS-sensors in an unmounted smartphone.

by

Frikkie Bruwer

Thesis presented in partial fulfilment of the requirements for the degree of Master of Engineering (Electronic) in the Faculty of Engineering at Stellenbosch University

Supervisor: Dr. M.J. Booysen

March 2017

The financial assistance of the Southern Africa Transport Conference (SATC) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the SATC.

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: March 2017 .....

Abstract

The field of vehicle telematics has been revolutionised by mobile- and sensor technology. Mobile phones have become pervasive and the increase in processing capacity, connectivity and richness of sensors have lead to dedicated vehicle telematics hardware being replaced by the widely obtainable and affordable smartphone.

The main inconveniences of using mobile phones for telematics sensing are battery-hungry systems (such as GPS) and app-related constraints on device position during operation (as required by systems using Microelectromechanical Systems (MEMS) sensors). Since users are in control of smartphones' permissions and application life cycles, these can be revoked and deactivated at will. Thus, to be a viable solution, the inconvenience caused to the user has to be minimised.

This thesis analyses and compares the efficacy and challenges of using the Global Positioning System (GPS) vs. Microelectromechanical Systems (MEMS) sensors. The main metric considered, apart from user convenience, is the ability to detect driving events. It is hypothesised that a generalised sensing platform can be developed to identify a complete and representative set of driving events by exclusively using MEMS sensors in an unmounted smartphone.

A system is developed for collecting, annotating and visualising driving data. More than 5000 user-identified driving events' data were collected and labelled using a developed, automated, Fuzzy Logic-based annotation algorithm.

MEMS sensor errors and errors induced by sampling from a smartphone-based platform were characterised and a thorough spectral analysis identified high frequency information in MEMS sensor data with specific relevance to driving event detection.

A unique multirate processing pipeline was developed using the acquired knowledge. By mitigating the identified errors, exploiting high frequency characteristics in the data and implementing a novel reorientation algorithm, the processing pipeline allows data from an unmounted smartphone to be transformed causally and efficiently into data apt for machine learning.

Hidden Markov Models and Random Forests are trained, tested and compared using the processed data. The results of the, better performing, Random Forests show a Balanced accuracy of 70.3% for classifying a complete set of 9 driving events at 2Hz and balanced accuracies of 93%, 88% and 78% for most prevalent events, turning, stationarity and coasting respectively.

Though the developed processing and classification systems have not been optimised, or implemented in the form of a smartphone application, the road has been paved for doing so. Given an application that can classify all significant types of driving events in a non-invasive and convenient way, information of specific relevance to an event of interest (i.e. yaw rate for a turn) could be parametrised and uploaded to the cloud. This could have a disruptive effect on the fields of vehicle telematics, intelligent transportation systems and participatory data aggregation.

Uittreksel

Die voertuig telematika vakgebied is revolutionêr verander deur selfoon- en sensor tegnologië. Selfone is vandag 'n alomteenwoordige verskynsel en 'n beduidende toename in die verwerkingskapasiteit, kommunikasiemediums en sensors-opsies het daartoe gelei dat toegewyde voertuig telematika hardeware, al meer vervang word deur slimfone.

Merkbare ongeriewe wat die gebruik van slimfone vir voertuig telematika veroorsaak, is: die batterylewe wat verkort word deur hoë drywing stelsels (soos Globale Posisioneringstelsel (GPS)); en beperkings op die toestel se posisie tydens die toep se werking (soos vereis deur stelsels wat van Mikroelektromeganiese stelsel (MEMS) sensore gebruik). Aangesien slimfoongebruikers volledig in beheer van toepassings se lewenssiklusse is, kan die toep gedeaktiveer word na willekeur. Dus, om 'n werkbare oplossing te word, moet die ongerief van slimfoonafhanklike voertuigtelematikatoepgebruik, tot die minimum beperk word.

Dit word in hierdie tesis gepostuleer dat 'n sagtewareplatform ontwikkel kan word, wat 'n volledige en verteenwoordigende stel bestuursgebeure kan identifiseer, deur uitsluitlik MEMS-sensore in 'n ongemonteerde slimfoon te gebruik.

'n Stelsel is ontwikkel vir die versameling, verwerking en visualisering van bestuursdata. Data van meer as 5000 – passasier geïdentifiseerde– bestuursgebeure is ingesamel en met behulp van 'n ontwikkelde, Fuzzy Logic-gebaseerde annoteringsalgoritme, van etikette voorsien.

Eienskappe van die MEMS-sensor foute wat veroorsaak word deur die slimfoon monsterproses is geïdentifiseer en maniere is gevind om dit teen te werk. 'n Deeglike spektrale-ontleding het hoëfrekwensie-inligting opgespoor wat aangewend kan word om bestuursgebeurtenisse uit te ken.

Met behulp van die verworwe kennis, is 'n unieke multitempoverwerkingspyplyn ontwikkel. Deur die geïdentifiseerde foute teen te werk, hoëfrekwensie-eienskappe te benut en die implementering van 'n unieke heroriënteringsalgoritme, kan die multitempoverwerkingspyplyn, data vanaf 'n ongemonteerde slimfoon kousaal en doeltreffend omskep na data gepas vir masjienleer.

Hidden Markov Models en Random Forests is opgelei, getoets en vergelyk met behulp van die ingesamelde-, geannoteerde- en verwerkte data. Die resultate van die toppresteerder, Random Forests, toon 'n *Balanced Accuracy* van 70.3 % vir die klassifikasie van 'n volledige stel van 9 bestuursgebeure teen 2Hz.

Hoewel die verwerking en klassifiseringstelsels nie geoptimeer is, of in die vorm van 'n slimfoontoep geïmplementeer is nie, is die fondasie wel neergelê om dit in die toekoms aan te pak. 'n Toep wat alle betekenisvolle tipes bestuursgebeure, op 'n gerieflike manier kan identifiseer, kan inligting wat van belang is vir elke bestuursgebeutenis ontgin en oplaai na die *Cloud*. Dit kan 'n dramatiese verbetering in velde van voertuigtelematika, intelligente vervoerstelsels en vrywillige dataversameling tot gevolg hê.

Contents

List of Figures

List of Tables

Nomenclature

Constants

g =	$9.8\,\mathrm{m.s^{-2}}$

Variables

v	Scalar speed	[$\mathrm{m.s^{-1}}$]
h	Heading	[rad]
g_y	Rotation rate around y-axis	[$\mathrm{rad.s^{-1}}$]
a_y	Acceleration in positive y-direction	[$\mathrm{m.s^{-2}}$]
q_b^a	Quaternion orientation of a w.r.t. b	[Unitless]
q_w	Scalar part of quaternion	[Unitless]
q_{xyz}	Complex part of quaternion	[Unitless]
θ	Rotation angle	[radians]
V	Alignment vector	[Unitless]
θ_z	Yaw	[radians]
θ_y	Pitch	[radians]
θ_x	Roll	[radians]
θ_{pre}	Pre-rotation	[radians]
θ_{tilt}	Tilt-rotation	[radians]
θ_{post}	Post-rotation	[radians]

Subscripts

$_x$	x-axis
$_y$	y-axis
$_z$	z-axis
$_{x'}$	Vehicle x-axis
$_{y'}$	Vehicle y-axis
$_{z'}$	Vehicle z-axis
$_s$	In the sensor axes
$_e$	In the earth axes
$_v$	In the vehicle axes
$_{est}$	estimated

Acronyms

ADAS	Advanced Driver-Assistance System

API Application Programming Interface
CAN Controller Area Network
CG Centre of Gravity
CSV Comma Seperated Value
DAS Driver-Assistance System
DCM Directional Cosine Matrix
DFT Discrete Fourier Transform
DTW Dynamic Time Warping
ECU Electronic Control Unit
EVD Eigenvalue Decomposition
FFT Fast Fourier Transform
FIFO First in First Out
FN False Negatives
FoM Figure of Merit
FP False Positives
GLOSA Green Light Optimal Speed Advisory
GNSS Global Navigation Satellite System
GPS Global Positioning System
GSM Global System for Mobile Communications
GUI Graphical User Interface
HF High Frequency
HMM Hidden Markov Model
HTML HyperText Markup Language
I^2C Inter-Integrated Circuit
IMU Inertial Measurement Unit
INS Inertial Navigation System
ITS Intelligent Transportation System
LF Low Frequency
LPF Low pass filter
M2M Machine to Machine
MEMS Michroelectromechanical Systems
NHTSA Unites States National Highway Traffic Safety Administration
OBD On-Board Diagnostics
OBU On-Board Unit
PCA Principle Component Analysis
RMS Square Route of Mean Squared
RTS Rauch-Tung-Striebel
SLERP Spherical Linear Interpolation
SMA Simple Moving Average
SPI Serial Peripheral Interface
SQL Structured Query Language

SVD Singular Value Decomposition
UBI Usage Based Insurance
WB Wheel Base
WHO World Health Organisation

Chapter 1

Introduction

Smartphones are being equipped with an increasingly rich set of sensing mechanisms. This can, in part be attributed to Microelectromechanical Systems (MEMS) sensors becoming smaller and more energy efficient. In combination, the MEMS sensors for measuring acceleration, rotation rate, magnetic field density, light and air pressure and other smartphone based systems such as the Global Positioning System (GPS) can provide a wealth of information previously only obtainable from expensive dedicated systems. To further enhance their utility, these smartphones boast a variety of connectivity mediums and rapidly increasing processing abilities.

The information and processing capabilities have been employed in a variety of applications, inclusing activity tracking. Using inertial sensors and GPS, smartphones can provide users with information on the number of steps taken or duration and distance of activities [1]. Due to their interconnectivity, smartphones are expected to play a significant role in the future of Machine to Machine (M2M) communications in vehicular networks [2].

According to the World Health Organisation (WHO), road injuries are the 9th leading cause of death worldwide and the leading cause of fatal injuries [3], this has lead to various vehicle manufacturers and other companies developing solutions to monitor a vehicles and driver behaviour [4, 5, 6].

The expensive nature of these systems and the lack of incentive for individuals to purchase them has lead to their primary use for vehicle fleet monitoring. However, the increasing prevalence of smartphones and their suitability for mobile sensing applications opens up a golden opportunity to easily implement vehicle monitoring systems on a large scale.

Smartphone-based sensing can- and has been successfully applied to the field of vehicle monitoring, where it has been used to monitor drivers' behaviour and inform them of potentially reckless manoeuvres they perform. Anonymous participatory sensing could facilitate a better understanding of scenarios leading up to accidents or point out problematic areas with high risks of accidents [7]. Equipped with an appropriate smartphone, public or private sector transport users can give an objective analysis of a driver's driving style and implement driver-ratings accordingly. Law enforcement could also benefit from updates on potentially dangerous drivers.

The omnipresent connectivity of smartphones also allows, by the user's consent, the implementation of other transport applications, such as traffic monitoring, traffic re-routing, accident reporting and large scale participatory data aggregation to cloud servers.

Smartphone-based sensing platforms are completely under the control of end-users,

who are responsible for installing, running and granting permissions for the application according to their preference. Various insurance companies entice users to activate the application by providing incentives in the form of discounts on monthly premiums and points systems that "gamify" safe driving [8, 9]. Further studies show that the accuracy of the information gathered and the concluding feedback to a driver has a notable effect on a user's confidence in a system [10]. A central aspect of smartphone-based vehicle monitoring that needs to be improved on in order to facilitate widespread adoption is therefore to make these applications as accurate and convenient (no constraints on phone position in vehicle and reasonable use of battery resources) as possible.

1.1 Problem statements

The resurgence of mobile - and sensor technology has had a disruptive effect on Intelligent Transport Systems (ITS) such as driver behaviour detection and vehicle monitoring. Due to the pervasiveness of mobile phones, their increasing processing capacity and their increasingly rich sensor set, enables shifting ITS applications to ubiquitously present smartphones. However, existing approaches are directly dependent on GPS technology for all data. There are various reasons why GPS dependence presents a problem. : 1. Limited accuracy; 2. Low sampling rate; 3. Slow signal acquisition; 4. Limited availability and 5. High power consumption.

An alternative to GPS for acquiring kinematic information is Microelectromechanical Systems (MEMS). However, MEMS sensors present their own set of challenging problems that need to be mitigated: 1. Mobility of the device within mobility of the vehicle; 2. sampling from a non-real-time operating system; 3. varying sensor quality; 4. limited processing power for signal processing.

1.2 Hypothesis

A generalised sensing platform can be developed to identify and describe driving events by exclusively using MEMS sensors in an unmounted smartphone.

1.3 Dissertation statements

1. Mobile phone based MEMS sensors are preferable to mobile phone based GPS in terms of accuracy and convenience when used to detect driving events.

2. A reorientation algorithm can be developed to allow MEMS sensor data to be rotated to a vehicle axis independent of sensor orientation within the vehicle for the purpose of driving event detection.

3. The gravitational acceleration measured by a smartphone-based MEMS accelerometer stationary within a vehicle, can be removed so it does not affect event recognition.

4. All significant driving events can be identified from smartphone-based MEMS sensor data by extracting high frequency characteristics and using machine learning algorithms.

5. The significance of individual sensor parameters for the classification of various driving events can be quantified.

1.4 Research objectives

A set of comprehensive research objectives were chosen in order to guide all research towards the deliverables necessary to prove the dissertation statements and, consequently, the hypothesis. The research objectives were:

1. To develop a complete smartphone-based system for collecting and annotating driving data.
2. To collect a raw GPS and MEMS sensor dataset with passenger annotations that is representative of general driving events, using smartphones.
3. Compare the efficacy of detecting driving events using GPS vs. MEMS sensors in terms of the accuracy and convenience of use.
4. To develop a tool for analysing and visualising collected data.
5. To develop a system that can process raw MEMS sensor data, from an unmounted smartphone, into a dataset that is relevant to vehicle dynamics, and that is compatible with machine learning methods.
6. To develop an algorithm that annotates the processed MEMS sensor dataset using collected passenger annotations and GPS data.
7. Compare the efficacy of suitable machine learning approaches in identifying driving events using the processed MEMS sensor dataset.

1.5 Original Contribution

The contribution made by this thesis can be subdivided into multiple novel contributions. These contributions support the development of a generalised sensing platform to identify and describe driving events by exclusively using MEMS sensors in an unmounted smartphone. The research produced a dataset consisting of raw smartphone-based MEMS sensor data, GPS data and multi-passenger annotations recorded from within a vehicle driving along a route chosen to be representative of general driving conditions. The research produced a reorientation algorithm, based on unique vehicle characteristics that can be detected using MEMS sensors. The algorithm allows MEMS sensor data to be rotated to a vehicle's axes independent of sensor orientation within the vehicle. The reorientation algorithm is evaluated using a distinct dataset with a rotating frame-of-reference data and an expected ground truth. An interface to the dataset, in the form of a web application with multiple tools for analysis, is contributed. A unique set of frequency spectra in smartphone-based MEMS sensor data is specified that have specific relevance to driving event identification. The research yields an algorithm that automatically annotates a MEMS sensor dataset using collected passenger annotations and GPS data. A unique method is contributed to process raw smartphone-based MEMS sensor data into a dataset that is relevant to vehicle dynamics, and that is compatible with machine learning methods. Subsequently,

a unique quantisation of the importance of individual variables for identifying driving events using machine learning will also be performed. The system is evaluated in terms of its ability to detect driving manoeuvres from smartphone-based MEMS sensor data.

1.6 Dissertation structure

The dissertation layout was chosen to ensure coherent grouping of information in chapters and a logical flow between the sequential parts with the goal of facilitating ease of reading. Figure 1.1 shows a visual representation of how the thesis statements relate to various chapters and sections in the thesis. The chapter following this introductory chapter, the literature review, is mainly concerned with the state of the art and identifying shortcomings, particularly in the field of smartphone based vehicle telematics. At the start of chapters 3 to 5, literature is given on methods used, and prior knowledge necessary to understand the subsequent chapter. Chapter 3 is mostly concerned with choosing the metrics to access what successful event detection entails and what the prerequisite conditions for evaluating successful event detection are. It also delves into the design of the experiment, including data capture scenarios and tools to be devised. Chapter 4 investigates the methodology used in the final system including the tools used for development, but excluding the methods employed to annotate and eventually detect driving events, which are discussed in chapter 5. The penultimate chapter, chapter 6 gives the result of evaluating the performance of algorithm from the two preceding chapters, chapter 4 and 5. The document is then concluded with a broader summary of what was achieved as well as deductions, implications, limitations and recommendations, in chapter 7.

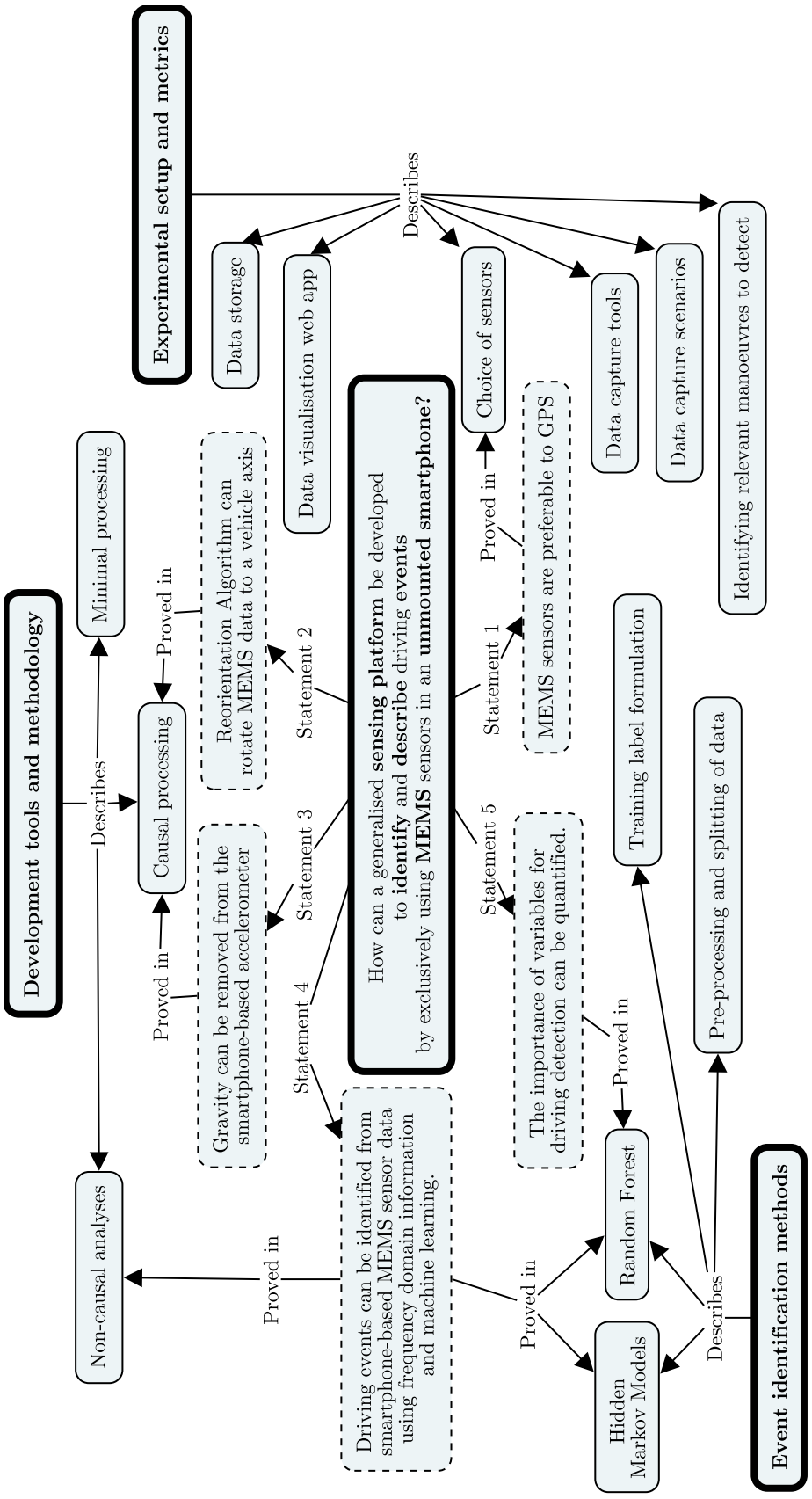

Figure 1.1: A concept map linking the statements in this proposal to the proposed structure of the thesis.

Chapter 2

Literature Review

The field of smartphone-based sensing has grown to such an extent that very strict boundaries were needed to limit the scope of the literature review. The focus of the review was exclusively on road-vehicle related sensing. Only sources considered to add a novel contribution were investigated. Where possible, special attention is given to:

- Type of vehicles
- Choice of sensors
- What is being detected (what is the output)
- Techniques (methods) used to achieve the output
- Testing methodology

Though the majority of projects discussed here are directly concerned with obtaining information using smartphone-based sensors, communication and interactivity between smartphones as a means to improve driving safety is also a pervasive theme and therefore the term Machine-to-machine (M2M) communications [2], will also make its appearance.

A thorough survey of smartphone based sensing in vehicles was published by Engelbrecht et al. [11] and contains more information related to the field.

The chapter will start by anatomising the field of smartphone based sensing in vehicles by identifying the main applications. A select set of ten entirely smartphone-based vehicle monitoring systems' functionality is described and summarised in section 2.2. Ultimately, a conclusion is formed regarding the most notable challenges facing smartphone-based vehicle monitoring systems

2.1 Smartphone-based sensing in vehicles

The existing literature on vehicle sensing can be categorised according to the four types of information that is captured, which is then disseminated in various ways for different applications. The types of data are:

- Traffic information, such as the location and movements of other vehicles or pedestrians

- Vehicle information, for example vehicle health telematics
- Environmental information, such as road conditions and weather conditions
- Driver behaviour information, including insurance telematics

Table 2.1 summarises a series of recent projects in the field of smartphone-based sensing in terms of the technology used, the type of vehicle sensing as well as the project's goal.

A very informative survey of the field was published by Wahlström et al. [12] and is a recommended read for more information on the history and current state of smartphone based vehicle telematics.

2.1.1 Traffic information

Smartphones in vehicles are being widely utilised for the aggregation and exploitation of traffic information. The automated detection of traffic jams and the consequent alternative route recommendations have been attempted by multiple mobile M2M systems [14, 15, 16, 37, 18]. An example is the Jam Eyes [18] system, which uses image processing and Wi-Fi to detect surrounding vehicles and, through collaboration with other smartphones running the system, calculate traffic jam lengths. Green Light Optimal Speed Advisory (GLOSA) is a term used to describe a system that can advise a driver of predicted traffic signal states and the optimal corresponding speed that would minimize braking and thereby improve fuel consumption. GLOSA is attempted by SignalGuru [20, 19], by identifying traffic signal light colours through smartphone-based cameras.

2.1.2 Vehicle information

Smartphone-based vehicle information monitoring is mainly used for vehicle diagnostics and other vehicle health related applications. Systems proposed by Zaldivar et al. [29] and Yang et al. [28] respectively, interface with a vehicle's electronic control unit (ECU) to retrieve and process information relevant to the vehicle operator and provide a user friendly interface.

2.1.3 Environmental information

Providing drivers and other transport users information w.r.t. the quality their transport route is another useful application for smartphone-based sensing and is of specific relevance for collaborative driving systems. Eriksson et al. [38], Mednis et al. [27] and Ghose et al. [26] have developed systems that detect road condition indicators such as potholes, bumps, rough- and uneven roads, using GPS and accelerometers. Fazeen et al. [24] experimented with road condition characterisation and mapping to a Google Earth map. The Nericell application developed by Trivedi et al. [25] is also designed to identify driving conditions such as potholes, bumps and honking at chaotic intersections.

Table 2.1: A table showing a summary of literature relevant to smartphone-based sensing in vehicles, adapted from a table by Engelbrecht et al. [11].

Ref.	Year	Technology	Category	Goal
[13]	2007	mobile phone	traffic information	mobile phones as sensor gateways
[14]	2012	smartphone, external sensors	vehicle information	engine parameters collection from external sensors
[15]	2013	smartphone, external sensors, ECU	vehicle information	opportunistic transfer of external sensor- and CAN bus data
[16]	2011	smartphone	traffic information	intelligent driver guidance tool
[17]	2012	smartphone	traffic information	road incident and traffic crowd-sourcing
[18]	2012	smartphone	traffic information	traffic queue length detection
[19] [20]	2011	smartphone	traffic information	traffic signal detection and learning
[21]	2009	smartphone	driver behaviour information	lane departure warning system
[7]	2011	smartphone	driver behaviour information	aggressive driving detection
[22]	2012	smartphone	driver behaviour information	driving style characterization
[23]	2010	smartphone	driver behaviour information	drunk driving detection
[24]	2012	smartphone	driver behaviour information, environmental information	advanced driver-assistance system
[25]	2008	smartphone	traffic information, driver behaviour information, environmental information	road and traffic condition monitoring
[26]	2012	smartphone	traffic information, environmental information	pothole detection and notification
[27]	2011	smartphone	traffic information, environmental information	pothole detection and notification
[28]	2013	smartphone	vehicle information	hybrid electric vehicle diagnostics
[29]	2011	smartphone, ECU	vehicle information	accident detection and notification
[30] [31]	2010	smartphone	vehicle information	accident detection and notification
[32]	2011	smartphone	driver behaviour information	eco-driving assistant
[33]	2012	smartphone	traffic information	eco-driving assistant
[34]	2014	smartphone	driver behaviour information	dangerous cornering detection
[35]	2014	smartphone	driver behaviour information	dangerous cornering detection
[10]	2014	smartphone	driver behaviour information	driving risk level scoring and driver feedback
[36]	2015	smartphone	driver behaviour information	fuzzy logic based driving risk level scoring

2.1.4 Driver behaviour information

In 2010 Dai et al. [23] developed a drunk driving detection system that aims at detecting drunk driving manoeuvres using only a smartphone-based accelerometer. In the following year Johnson and Trivedi [7] developed a smartphone-based driver behaviour monitoring system that attempts to classify several aggressive and non-aggressive driving manoeuvres using only smartphone-based sensors (MEMS-sensors and GPS).

DriveGain [39] is an example of an application that attempts to improve driver fuel economy by advising the driver of bad driving habits according to the detected driving style.

Vehicle insurance telematics aids insurance companies in quantifying the actuarial risk associated with a driver. Händel et al. [10] provides a detailed investigation into the necessary characteristics of a smartphone-based insurance telematics system.

Wahlström et al.[35, 34], developed two systems that detect and quantify the risk level of cornering events using GPS measurements. These risk classifications are used for data driven insurance telematics.

Castignani et al. [36] proposed the SenseFleet system for driver behaviour monitoring. A novel, vehicle specific calibration procedure is used to determine fuzzy logic sets that enable the distinction between calm and aggressive driving to be made. Driver behaviour is therefore identified independent of the vehicle's characteristics.

2.2 Entirely smartphone-based vehicle monitoring systems

The advent of mobile technology has meant that many of the driver behavioural applications mentioned in Section 2.1 can use smartphone-based sensors. The use of smartphone-based sensing as opposed to sensing from a fixed installation in a vehicle, has certain advantages that include

- links behaviour to an individual, rather than to a vehicle of which the driver might be unknown
- decouples the vehicle's age, technology, make, type, and interfaces from the sensing solution
- provides connectivity without any additional equipment
- negates installation costs
- enables the detection of phone usage while driving.

Despite these benefits, the use of only smartphone-based sensing introduces many challenges too, including

- sensor and algorithm complexity vs. limited battery power
- the changing orientation of the devices which makes vehicle acceleration sensing cumbersome
- broad range of smartphones
- inaccuracies in cost-effective sensors used in smartphones.

Table 2.2: Summary of techniques, hardware, software, and sensors used by smartphone-based vehicle monitoring systems.

Reference	Hardware	Software	Detection technique	Sensors used
Trivedi [25]	HP iPAQ hw6965 PDA, HTC Typhoon smartphone, Sparkfun WiTilt accelerometer	Windows Mobile 5.0 and 2003	pattern matching, orientation calibration	accelerometer, microphone, GPS
Dai [23]	HTC Dream (G1) smartphone	Android 1.6	pattern matching, orientation calibration	accelerometer, gyroscope
Johnson [7]	iPhone 4	iOS 4	endpoint detection, DTW	accelerometer, gyroscope, magnetometer, GPS
Eren [22]	iPhone 4	iOS 4	endpoint detection, DTW, Bayesian classifier	accelerometer, gyroscope, magnetometer
Fazeen [24]	HTC Google Nexus One smartphone	Android 2.1	pattern recognition	accelerometer, GPS
White [30]	HTC Magic (Google ION) smartphone	Android 1.5	pattern matching	accelerometer, microphone, GPS
Wahlström [35]	Samsung Galaxy S3, Galaxy Xcover 2, and Galaxy S4	unspecified	theoretical tyre slip threshold detection	GPS
Wahlström [34]	iPhone 4, iPhone 5 and Samsung Galaxy S5	unspecified	theoretical tyre slip threshold detection	GPS
Händel [10]	HTC Desire HD, iPhone 4, iPhone 5 and Samsung Galaxy S3	unspecified	multiple FoMs threshold detection	GPS
Castignani [36]	Samsung Galaxy Gio and a Samsung Galaxy S3	unspecified	fuzzy logic	GPS, accelerometer, magnetometer, weather, time of day

These challenges demonstrate why the focus of this literature review is purely smartphone-based solutions will be analysed in more detail in the rest of the literature review, starting with table 2.2, showing a summary of the hardware, software, detection techniques and choices of sensors used in a selected set of smartphone based systems in the literature.

The conventions for the axes and units of information used to describe vehicle- and smartphone kinematics are as follows : Vector components in the smartphone axes are described as x, y and z, corresponding (on Android and iOS operating systems) to the right of the phone, the top of the phone and the screen of the phone respectively. When denoting the vehicle axes a accent is added and x', y' and z' describes the front, left and top of the vehicle respectively. These conventions are illustrated in Figure 2.1. These

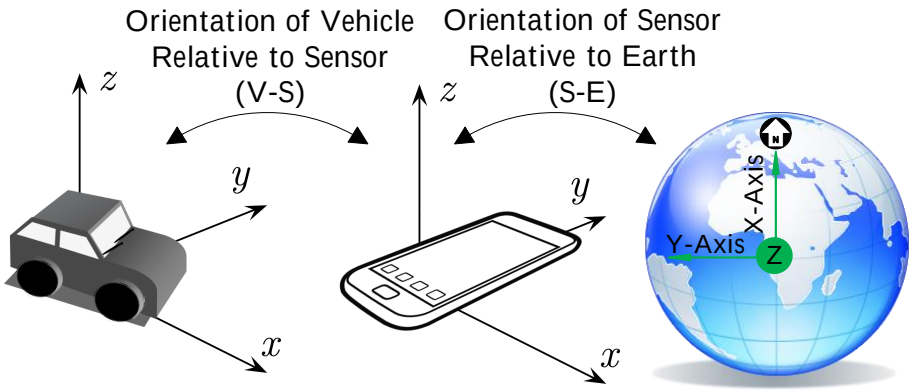

Figure 2.1: This figure shows the conventions used for the sensor- vehicle- and earth axes as well as the rotations between the axes.

Table 2.3: Summary of conventions used for x-,y- and z-directions in the earth-, vehicle- and sensor-axes

Axes	Direction		
	X	Y	Z
Earth	Magnetic North	Magnetic West	Upwards from surface
Vehicle	Front of vehicle	Left of vehicle	Top of vehicle
Smartphone	Right hand side facing screen	Top of device when facing screen	Pointing out of screen

axes descriptions are used as subscripts to indicate the acceleration or rotation rate in a specific axis, i.e. a_x and ω'_y for acceleration in the direction of the smartphone's positive x-axis and rotation rate around the vehicle's positive y axis respectively. These conventions are also summarised in section 4.3.4.

2.2.1 Nericell: Rich Monitoring of Road and Traffic Conditions using Mobile Smartphones — Trivedi et al.

Nericell is a smartphone application that attempts to detect vehicle braking, traversing a bumpy road, horns of vehicles in the vicinity and stop and go traffic, with the goal of annotating traffic maps with rich data w.r.t. traffic conditions and road quality. The application employs the smartphone-based accelerometer, microphone, GPS as well as the GSM based localisation service. The application makes progress towards a reasonable power consumption by proposing to constantly monitor the accelerometer and only use the GPS and microphone (in combination more than 800mW compared to the less than 5mW of the accelerometer) when necessary.

2.2.1.1 Virtual reorientation procedure

Two collected vectors are used to generate 3 Euler rotation angles used for rotating accelerometer data to the vehicle axes. An estimation of the vehicle's vertical axis is obtained from the median accelerometer vector, which is assumed to correspond to the top of the vehicle. The vector representing the rear of the vehicle is obtained by monitoring the GPS for significant-, straight-line decelerations and recording the accelerometer value during the deceleration. Two obvious drawbacks of these methods

are that: 1. When the vehicle is not on a perfectly level road, the gravity vector does not correspond to the vertical axis of the vehicle. 2. Waiting for a significant straight-line deceleration can delay the completion of the reorientation procedure.

The reorientation algorithm is evaluated by comparing the cross correlation of two well aligned accelerometers with the cross correlation between the well aligned accelerometers and virtually reoriented data from a malaligned accelerometer.

2.2.1.2 Event detection

Events are identified using simple thresholding techniques applied to the accelerometer and sound data. Algorithm performance is evaluated by comparing classifications to manual annotations as well as annotations generated by heuristics from GPS-speed.

2.2.2 Mobile Phone-Based Drunk Driving Detection — Dai et al.

Dai et al. [23] attempts to detect manoeuvres directly associated with drunk driving using a mobile phone. The paper refers to a publication by the Unites States National Highway Traffic Safety Administration (NHTSA) [40]. The study discusses drunk driving related behaviours and highlights two categories of problematic symptoms exhibited by drunk drivers: 1. Lateral positioning problems such as weaving, drifting and swerving. 2. Speed regulation becomes an issue and symptoms, such as harsh accelerations or irregular braking, are observed. These symptoms can all be observed from longitudinal and lateral acceleration.

2.2.2.1 Reorientation Procedure

The reorientation of the accelerometer data is done by calculating only acceleration components $a_{x'}$ and $a_{y'}$ representing the vehicle's lateral and longitudinal acceleration respectively. The system developed by Dai et al. calculates the horizontal components of a_x and a_y (the system ignores a_z for some unmentioned reason) from the angles provided by the Android orientation sensor using Equation 2.1.

$$\begin{aligned} \mathbf{a}_{xh} &= \mathbf{a}_x \cos(\theta_y) \\ \mathbf{a}_{yh} &= \mathbf{a}_y \cos(\theta_x) \end{aligned} \tag{2.1}$$

The orientation sensor [41] is a virtual sensor programmed into certain smartphones' operating systems. It uses proprietary algorithms to estimate a device's orientation with respect to gravity and magnetic north, usually from accelerometer- and magnetometer—in some cases gyroscopic– readings.

These horizontal acceleration values are aligned to the vehicle's longitudinal and lateral axes by using the first acceleration above a predefined threshold detected. A method is implemented to avoid incorrect calibration in a case where the first significant acceleration is in a reverse direction. The method involves inverting the direction of the estimated front of the vehicle if the subsequent acceleration is in the opposite direction to the first. This method, however, would also reverse the direction of the vector if an acceleration followed by a brake takes place and therefore makes no sense.

The system assumes the accuracy of the virtual orientation sensor, this sensor, however cannot correctly determine device orientation relative to earth when the device experiences a significant acceleration and a significant acceleration is required to

determine the front of the vehicle. Secondly, only the horizontal components of acceleration are used by the system, therefore any incline encountered during operation would result in inaccurate accelerometer readings. The orientation of the device relative to the vehicle must also remain constant throughout operation, as calibration is only performed once at start-up.

No conclusive results are shown for the calibration procedure.

2.2.2.2 Pattern Matching

The system developed by Dai et al. implements, what they call, a pattern matching system for identifying the symptoms of drunk driving. After further investigation, however, it was found that their definition of a pattern matching only involves calculating the difference between the maximum and minimum values of a'_y occurring in a fixed length window. All event types were thus detected by simply comparing accelerometer values to empirically defined thresholds within a 5 second window period. To add robustness to the classifications, "multiple round pattern matching" is performed, by performing the same "pattern matching" on consecutive windowed periods and only accepting an event classification if the criteria for the event is met for several consecutive windows.

The algorithms were tested using 72 sets of data containing drunk driving related events and 22 sets of data representing "regular" driving behaviour. Surprisingly good results are claimed, given the simplicity of the algorithm. Low (less than 3%) FP and 0% FN rates are reported after adjusting the thresholds to obtain the best balanced accuracy. Only a slight, acceptable decrease in accuracy (less than 15% FN and 3% FP) is claimed when the phone is sliding during a turn, despite the reorientation algorithm only being performed once at the start of driving.

The system does not take into account the effects of gravity on the accelerometer, so performance should deteriorate on non-level road surfaces. Calibration (calculation of angles for reorienting accelerometer data) is only performed once, requiring the phone to remain in the same orientation w.r.t. the vehicle throughout the trip. No suggestion is given as to how the data was annotated for the testing procedure.

2.2.3 Driving Style Recognition Using a Smartphone as a Sensor Platform — Johnson and Trivedi

In 2011, Johnson and Trivedi [7] developed the MIROAD system, which attempts the recognition of a set of driving events using a smartphone-based accelerometer, gyroscope and GPS in a fixed mount aligned with the vehicle axes. The events for which detection are discussed and evaluated are:

1. left/right aggressive/non-aggressive turns
2. aggressive/non-aggressive U-turns
3. left/right Swerves

Additional events, such as aggressive braking, removal of the phone from its mount and excessive speed are claimed to be detected, but no results or methodology are given.

The MIROAD application detects the start and endpoints of turning events by using a simple moving average (SMA)of the vehicle's yaw rate ($\omega_{z'}$). When the SMA exceeds an empirically determined threshold, the event start is defined and when it falls below another threshold or exceeds 15 seconds, the end of the event is defined. This segmentation technique allows the system to only perform further processing when an event is expected to be occurring.

Dynamic Time Warping (DTW), a technique for obtaining a dissimilarity measure of two differing length non-aligned time series, can now be used on the segmented event data by comparing data from the event time-frames to pre-defined templates representing the manoeuvres to be detected. The template showing the highest similarity to the segment of data determines the classification. The data from the accelerometer, gyroscope and Android's orientation sensor are used in different combinations to form the templates and the data for comparison.

A set of 201 driving events with three different drivers and vehicles was collected and used to test the classification accuracy. The method used to collect and annotate the dataset is not described.

The highest recognition rates were achieved by using a 3 part vector consisting of the lateral acceleration ($a_{y'}$), yaw rate ($\omega_{z'}$) and the yaw angle ($\theta_{z'}$) w.r.t. the reference attitude (the attitude of the phone at the start of the event).

Results showed an average detection rate of 91 % with the worst performing detection being the U-turn at 77%, though no mention is made of the detection rate of the SMA segmentation method employed.

This system is an interesting hybrid of an event recognition system and an event classification system as it detects both aggressive and non-aggressive turns and U-turns. Johnson and Trivedi, however, recommend that the binary aggressive or non-aggressive lable be used to tally the amount of aggressive events in an area in order to give insight into where incidents are likely to occur. A significant limitation of MIROAD is its operational boundaries. For the application to function correctly, the smartphone must be stationary and aligned with the vehicle axes. The system also consumes more power than other systems, due to the computationally intensive DTW algorithm and the permanent use of GPS.

2.2.4 Estimating Driver Behaviour by a Smartphone — Eren et al.

Eren et al. [22] proposes a system that uses methods similar to that of MIROAD [7](an endpoint detection algorithm to segment data and a DTW algorithm comparing the segmented data to templates), but with an additional Bayesian classifier implemented in order to reach a different end objective. The objective of the system is to simply classify a driver's behaviour as *safe* or *risky*. This is a significantly easier target to achieve for multiple reasons. Firstly, of the 10 drivers' data used testing, 2 where determined, by their subjective manual annotation, to show risky driving behaviour. Despite these 10 drivers' data being used as a testing set, the prior probability of risky driving was set to 0.2, though real world figures should be significantly lower. Secondly, a full 5 minute dataset containing multiple events could be used to reach the conclusion regarding each drivers behaviour compared to other systems where each event is individually classified.

The results show that, of the 15 drivers' being classified (apparently all data was used for the training and testing sets), 14 were correctly classified.

Though the results of the system looks promising, it is difficult to assess the effectiveness of the techniques used as the binary classification attempted is significantly easier than the event classification attempted by other systems in the literature. Having to place the smartphone in a fixed position within the vehicle, and aligned with its axes, is also a significant limitation of the system.

2.2.5 Safe Driving Using Mobile Phones — Fazeen et al.

The advanced driver-assistance system (ADAS) system conceptualised by Fazeen et al. [24], uses data from an accelerometer to inform a driver of vehicle condition (by correlating detected gear shifts with speed obtained by integrating the accelerometer), Driving patterns (identifying when longitudinal or lateral accelerations exceed safe thresholds) or road conditions (identified from vertical axis acceleration patterns and integrated accelerometer speed).

The vehicle condition information was not explicitly calculated, but it was shown that gear shifts could be manually identified from the accelerometer data and that the accelerometer can be integrated over to produce a reasonable accurate speed approximation (a maximum error of 4.14% is described).

The identification of driving patterns was also not performed by a developed automated algorithm, however, a brief quantitative analysis of the patterns and thresholds relating to safe and "sudden" accelerations, braking and lane changes are given.

The mapping of road condition is done using a "dynamic classification method" of which no details are given. Every GPS sample collected is classified as one of:

- Bump
- Pothole
- Rough
- Smooth
- Uneven

These samples are then colour coded and plotted to a Google Earth map, giving an indication of the road condition for a city.

Classification accuracy is given as between 72% for potholes and 92% for smooth road, but classification is assumed to be manually derived from visual sensor data and no details are given with regards to the ground truth data used to test classification accuracy. All data is recorded with the smartphone in a fixed position aligned with the vehicle axes. No battery drain information is given, but due to the exclusive use of accelerometer data for detection (excluding the GPS coordinates used for road condition mapping) and the lack of computationally expensive algorithms, the system is considered very efficient.

2.2.6 WreckWatch: Automatic Traffic Accident Detection and Notification with Smartphones — White et al.

The WreckWatch [30] system differs significantly from the other systems discussed in this literature review in terms of what is detected as well as how detection is attempted. The system takes a holistic approach to developing a feasible method for detecting accidents using smartphone based sensors, reporting accidents to relevant authorities, providing contextual information and opening a communication channel between the victims and first responders. The most relevant information w.r.t. this thesis is the detection of events using smartphone-based sensors.

Information collected from the GPS, accelerometer and microphone are combined using a complex rule-based system to reach a consensus that minimises the chances of a false positive occurring. The limitations of the various sensors are identified, for example: The accelerometer reading is susceptible to false positives caused by the device's motion within the vehicle. The device could be dropped, causing a large accelerometer spike and, consequently, a false positive. The sound recorded by the microphone is clipped at 150 dB, this volume level could be achieved using the vehicle's stereo, causing a false positive. By using all information available, however, the probability of a False Positive is dramatically reduced.

A few significant shortcomings of the developed smartphone-based system are, however, pointed out in the paper:

- The high battery consumption of smartphone-based GPS and audio processing might discourage users from activating the application.
- Vehicle comfort and safety systems are designed to limit peak accelerations experienced within the vehicle, this dampening of acceleration can remove usefull information from the accelerometer reading, especially compared to proprietary-hardware systems mounted to the frame of the vehicle.
- All smartphone-based applications run within confines of their operating systems, which, in the case of the two most popular mobile operating systems, are in some scenarios quite narrow.

2.2.7 Detection of Dangerous Cornering in GNSS Data Driven Insurance Telematics – Wahlström et al.

2.2.7.1 Quantifying Dangerous Cornering

The two main dangers presented during harsh cornering are tyre slippage(slipping) and vehicle rollover. With this in mind, Wahlström et al.[35] defined a dangerous cornering event relative to the thresholds where slipping or vehicle rollover are theoretically likely to occur. The theoretical slipping threshold is defined with respect to the tangential velocity(v), rotational velocity (w) and tangential acceleration(a) of the vehicle as well as the coefficient of friction (μ) between the vehicle's tyres and the road. To avoid slipping, inequality 2.3 must hold. A function $T(v, a, w)$ is defined as the ratio between the horizontal F_h and normal F_n forces exerted by the road on the vehicle.

$$T(v, a, w) = \frac{F_h}{F_n} = \frac{1}{g}\sqrt{v^2w^2 + a^2} \tag{2.2}$$

$$T(v, a, w) \leq c1\mu \tag{2.3}$$

$$c1\mu = \gamma_{ns} \tag{2.4}$$

Where $c1$ is a constant safety factor chosen empirically and γ_{ns} is the adjusted no slip threshold.

For the theoretical rollover threshold, the torque around the vehicle's centre of gravity(CG) must be considered. The vehicle's track width (l) as well as the height of the CG (h) need also be taken into account. It can then be proven that, to avoid rollover, inequality 2.5 must hold.

$$T(v, 0, w) \leq c2\frac{l}{2h} \tag{2.5}$$

$$c2\frac{l}{2h} = \gamma_{nr} \tag{2.6}$$

Where $c2$ is a constant safety factor chosen empirically and γ_{nr} is the adjusted no roll threshold.

Evaluating these equations for various vehicles it can be shown that, for a typical case with no additional roof loads, zero road pitch- and roll angles and no tripped rollovers, the no-slip threshold will be reached before the no-rollover threshold. Wahlström et al defines a single cornering event from the moment where $T(v, a, w) \geq \gamma_{ns}$ to the moment where $T(v, a, w) \leq \gamma_{ns}$. However, when $T(v, a, w) \geq 0.35$ (0.35 being empirically chosen) between two events, they are combined as a single cornering event. The risk level of the event is defined as the maximum value of $T(v, a, w)$ during the event.

2.2.7.2 Filtering of GNSS Measurements

A number of input variables are needed to enable the use of the methods in Section 2.2.7.1. These variables need to be obtained from Global Navigation Satellite System (GNSS) measurements, according to Wahlström et al. however, the GNSS receivers found in smartphones are not of very high quality and the data they provide is prone to errors. The variables are therefore estimated using a Kalman Filter.

Most mobile phone based GNSS systems supply at least the two dimensional coordinate position as well as the Doppler-based heading and velocity of the device. To improve the accuracy of the estimated variables, the position measurement is modeled as a combination of discrete, zero-mean white noise as well as a slowly varying bias caused by clock errors and atmospheric effects. The slow moving bias is effectively removed by using

$$\Delta\mathbf{p_n} = \mathbf{p_n} - \mathbf{p_{n-1}} \tag{2.7}$$

instead of the position measurement for calculations. Due to the non-linear nature of the equations governing the system's dynamics and its ability to apply constraints to individual sigma points, an Unscented Kalman Filter is used to estimate the required variables. The filter's parameters such as system and measurement noise covariances are chosen to correspond to typical vehicle dynamics as well as expected GNSS error magnitudes.

A framework for parameter tuning and performance evaluation is described, where events detected by the system are mapped to events provided by a reference system.

Various options for loss functions based on the number of missed and false detections are given and certain parameters can be modified to minimise the loss function.

2.2.7.3 Testing and Results

The system was field tested using 31 minutes of aggressive driving data. A Microstrain 3DM-GX3-35 was used to collect reference data using its IMU and GNSS sensors. The IMU and GNSS sensor data was combined to provide a reverence value for $T(v, a, w)$. Data was simultaneously collected from three different Android smartphones. Thresholds for γ_{SL} were chosen between 0.5 and 0.75, according to Wahlström et al. these values correspond to what should intuitively be labeled as aggressive cornering.

For thresholds in the range of $0.5 \leq \gamma_{SL} \leq 0.6$ The system showed a average 40% missed detection and false alarm rate over the three phones used. This percentage deteriorates for higher thresholds. The estimator is also less effective when dealing with high g-force cornering, due to these events typically having a shorter duration and the GNSS update rate being too slow.

The novel and useful framework for parameter tuning can be better utilised by applying large amounts of driving data, as the 31 minutes of driving data used cannot be seen as a fully representative sample. The advantage of using only GNSS data is that it is independent of device orientation and movement relative to the vehicle. GNSS on a smartphone, however, has a high power consumption, slow update rate (typically around 4Hz) and is often unreliable in terms of satellite signal and accuracy.

Fusing readily available smartphone IMU data with GNSS data could increase system reliability and the higher update rate of the IMU sensors could aid in detecting shorter high g-force events.

2.2.8 Risk Assessment of Vehicle Cornering Events in GNSS Data Driven Insurance Telematics – Wahlström et al.

Similar to the other paper presented by Wahlström et al.[35], theoretical thresholds for rollover and slip are calculated and used as a reference for assessing the risk level of cornering events, but instead of $T(v, a, w)$ in Equation 2.2, $T(v, a, r)$ is used, where r is the radius of the specified turn. This leads to Equation 2.8.

$$T(v, a, r) = \frac{F_h}{F_n} = \frac{1}{g}\sqrt{\frac{v^4}{r^2} + a^2} \tag{2.8}$$

The radius r is estimated using a circle fitting technique to fit circles to position measurements. The velocity (v) and acceleration (a) parameters are estimated from the GNSS measurements using a Rauch-Tung-Striebel (RTS) Kalman smoother.

The system functionality was first demonstrated by simulation with generated data. A field study was then conducted with three smartphones (iPhone 4,iPhone 5 and Galaxy S5) collecting GNSS data simultaneously. The differences in smartphone position measurements was expected to have a limited effect on the radius estimates, but the one smartphone(iPhone 5) produced an outlier data point resulting in a false radius estimation and increased risk. No attempt was made to compare the smartphone data with data from a reference system, therefore, despite the theoretical merit of the system no comparison in terms of accuracy can be made.

2.2.9 Insurance Telematics: Opportunities and Challenges with the Smartphone Solution – Händel et al.

The article presented by Händel et al. [10] does not aim at the development and testing of a complete smartphone-based reckless driving detection system, but rather discusses the technological aspects of vehicle insurance telematics and the use of smartphones as a platform for vehicle insurance telematics.

Eight of the most common Figure of Merits (FoMs) that can be used for insurance telematics are identified and characterised in terms of a number of quality measures, namely: observability, event stationarity, actuarial relevance and driver influence. Observability indicates how accurately the figure of merit can be derived from the sensor measurements. Stationarity is defined as the time length in which the measurements that relate to a FoM can be measured. Actuarial relevance indicates how indicative the event is of the risk-level of insuring the driver. Driver influence indicates to what degree the driver has an effect on the FoM. The characterisation of the 8 FoMs, if obtained from GNSS-measurements, are shown in Table 2.4.

Table 2.4: Characterisation of FoMs in insurance telematics when calculated using gnss-data. [10]

FoM	**Description**	**Observability**	**Stationarity**	**Driver-Influence**	**Actuarial Relevance**
Acceleration	Number of rapid acceleration events and their harshness	Medium	Low	High	Medium
Braking	Number of harsh braking events and their harshness	Medium	Low	Medium	High
Speeding (Absolute)	Amount of absolute speeding	High	High	High	Medium
Speeding (Relative)	Amount of speeding relative a location dependent limit	Medium	High	High	High
Smoothness	Long-term speed variations around a nominal speed	High	High	Medium	Low
Swerving	Number of abrupt steering maneuvers and their harshness	Low	Low	Medium	Low
Cornering	Number of events when turning at too high speed and their harshness	Medium	Medium	High	Medium
Eco-ness	Instantaneous or trip-based energy consumption or carbon footprint	Low	Medium	High	Low
Elapsed time	Time duration of the trip	High	High	Low	Low
Elapsed distance	Distance of the trip	High	High	Low	High
Time of day	Actual time of day when making the trip	High	High	Low	High
Location	Geographical location of the trip	High	High	Low	Medium

An alternative method to direct differentiation(which amplifies high frequency noise) of GNSS speed data is proposed, to obtain "clean" acceleration data as well

as a quality index indicating the quality of the calculated data. A test of the percentage GNSS coverage and harsh breaking detection ability of 7 popular smartphones was done by using a vehicle's OBD data as a reference. The test confirmed the unreliability of the directly differentiated smartphone data and showed that the low-level data cleansing routine decreased false detections.

A scoring system for individual FoMs is developed. A score between 0 (bad) and 1 (good) is allocated using Equation 2.9.

$$\mathbf{S} = \frac{1}{1 + \alpha f} \tag{2.9}$$

With f denoting the number of detected events and α being a chosen constant. Given a statistical distribution of f, modeled using the information in Table 2.4 one can calculate an appropriate value of α for each FoM.

To combine scores for all FoMs, the scores can be scaled according to their actuarial relevance. A trade off has to be made between a reliable system which places less weight on FoMs with low observability and stationarity versus placing emphasis on actuarially relevant FoMs. The relevance of the scoring feedback to the driver must also be carefully evaluated, because actuarially relevant data such as day or nighttime driving is out of the drivers control and low scores due to such FoMs could cause drivers to lose confidence in the system.

2.2.10 Driver behaviour Profiling Using Smartphones – Castignani et al.

The SenseFleet [36] system proposed by Castignani et al. uses smartphone-based accelerometer, magnetometer and GPS sensors to identify and rate the riskiness of driving events. This is done by defining Fuzzy logic sets for each parameter after a calibration phase in a specific vehicle. The relevant parameters chosen are the standard deviation of the jerk (derivative of accelerometer measurement), mean yaw rate(using orientation sensors) as well as the speed and bearing variations(from GPS measurements). Calibration is vehicle-specific, as different vehicles have unique driving characteristics. Low, medium high and very-high values are calibrated for all parameters except speed variation, for which HIGH-DEC, LOW-DEC, STABLE, LOW-ACC and HIGH-ACC values are defined. Detection can be done using a basic fuzzy inference system. Weather and time of day data are also gathered for each event so driver scoring can be adjusted accordingly. A simple scoring system is implemented where a driver starts a trip with 100 points and a predefined amount of points are subtracted for each type of event and the conditions during which the event takes place(i.e. more points will be subtracted for an event happening at night in snow than in the daylight in sunny weather.). Points are added when no events are detected for a set interval.

The calibration method proposed, requires the user to drive the vehicle while a set amount of samples are collected within each of three speed intervals. The necessity of this cumbersome vehicle specific calibration is a notable disadvantage of the system.

To experimentally test the system, ten drivers were asked to, after the calibration phase, drive a predefined route twice. The first lap was driven calmly and the second lap aggressively. The results showed an average driver score of 74.4 for the calm lap and 20.3 for the aggressive lap. A Principle Component Analysis (PCA) was done on the data and the resulting data could easily be clustered according to the aggressive

and calm laps. The system therefore allows clear distinctions to be made between aggressive and calm driving behaviour. A test was also executed using a high end and entry level Android smartphone mounted in a similar way within the vehicle. The two phones showed similar results, demonstrating the compatibility of the system with different smartphone models.

2.3 Challenges facing smartphone-based vehicle monitoring systems

The literature reviewed in section 2.2 highlights a set of significant challenges that need to be addressed thoroughly before smartphone-based vehicle monitoring can become a widely adopted vehicle monitoring method. The challenges of Automation, Operational boundaries, Power consumption and Data aggregation are all– to some extent – addressed in this thesis.

2.3.1 Automation

Driving is an activity that demands as much as possible of a person's concentration to improve its safety. Any application demanding even a meagre percentage of the driver's concentration or interaction is considered dangerous and contrary to the goals of most of the mentioned systems. Compulsory calibration routines, obtrusive visual notifications and any required interaction aside from activating the service must be avoided at all costs.

2.3.2 Operational boundaries

Many of the existing applications, such as the MIROAD system by Johnson and Trivedi [7] or the system developed by Eren et al. [22], only work when used within strict operational boundaries (i.e. the phone must be mounted in a fixed position, aligned with the vehicle axes). This can cause significant inconvenience when the user does not own a suitable mounting system and completely prevents the application from being used in public or informal transport scenarios. An ideal system should be usable from any orientation within the vehicle and application operation cannot be inhibited if the device orientation changes periodically within the vehicle.

2.3.3 Power consumption

A smartphone application's life cycle is completely under the control of the user. If an application inconveniently drains a significant percentage of a user's battery during operation, it is fully within the user's ability and rights to turn off the application. It is therefore imperative that smartphone-based vehicle monitoring applications are designed to use power hungry sensors, such as GPS, sparingly and to minimise computationally expensive processing algorithms such as those used for audio and visual processing.

2.3.4 Data aggregation

With the recent worldwide emphasis on the collection and processing of *Big Data*, smartphones have been implemented to collect information w.r.t. user location, activities, device usage, etc. It is inevitable that existing smartphone-based information aggregation capabilities will be expanded to collect data relating to vehicles and vehicle activity as well. However, just as step-counting is an important feature for collecting information on a user's physical activity, driving event identification will be a necessary tool for aggregating rich information with limited bandwidth. Large, comprehensive datasets are a necessity for developing algorithms to perform driver behaviour- , vehicle information-, environmental information- and traffic information monitoring. And, as pointed out by Trivedi et al.[25], aggregated data relating to road conditions and traffic information could aid in identifying high risk driving areas.

2.3.5 Wide-scale deployment

One of the notable reasons why vehicle monitoring has not achieved a deeper market penetration is the reluctance of vehicle owners to buy expensive dedicated hardware systems. Additional hardware not only increases costs, but also makes acquisition and installation cumbersome and inefficient, while a mobile phone application can be downloaded within seconds and with no cost to the user. To facilitate adoption by road users, insurance companies [42] have begun to provide discounts on premiums for users who allow themselves to be monitored. The monitoring allows the companies to adjust premiums according to the actuarial risk associated with a driver or vehicle.

Chapter 3

Experimental setup and metrics

3.1 Introduction

This chapter describes the design of a system for acquiring data, that can be utilised for vehicle monitoring or reckless driving detection, from a smartphone-based platform. This includes the processes used to:

- Identify the most relevant manoeuvres to detect.
- Choose the most suitable sensors (in terms of efficiency, accuracy, etc.)
- Design testing scenarios to enable effective data aggregation
- Implement applications to facilitate the gathering of data.
- Store data using a method that facilitates ease of access.
- Analyse the data for informative and applicable content

Methods described in this chapter are designed to facilitate progress towards research objectives 1 through 4 in Section 1.4. Figure 3.1 gives an overview of the complete system's dependency's and shows the chronological (left to right) development process.

The choice of relevant manoeuvres to identify and what data could be used to detect them is described in Section 3.2. Given a set of manoeuvres to detect, the choice of sensors to obtain the data is detailed in Section 3.3. Representative and informative driving scenarios in which data should be collected and the method used to collect the chosen sensor data is described in Section 3.4, with the procedure for collecting the annotated training data and collecting data to enable the development of the vehicle-sensor reorientation algorithm. The development of the logging and flagging applications that are used to collect and annotate the data is detailed in Section 3.5. Finally, the storage scheme and tools developed to aid in the analysis of data are discussed in sections 3.6 and 3.7.

3.2 Identifying relevant manoeuvres to detect

Händel et al., in their thorough study of insurance telematics, identified a set of 12 Figures of Merit (FoMs) and classified them in terms of observability, event stationarity,

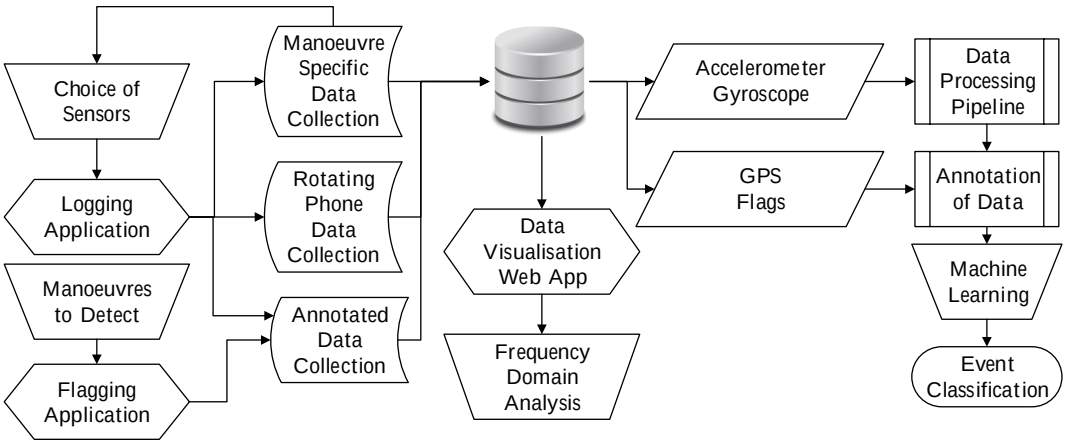

Figure 3.1: An information-flow diagram of the complete developed system.

actuarial relevance, and driver influence [10]. Observability is defined as the accuracy with which the FoM can be recorded using GPS, stationarity evaluates for how long a timeframe the FoM retains its observability, driver influence qualifies to what extent the driver has the ability to change the FoM, and actuarial relevance reflects the amount of risk caused by the FoM.

Table 3.1 shows the characterisation of selected FoMs by Händel et al. [10]:

Table 3.1: Characterisation of FoMs in insurance telematics when calculated using GPS data by Händel et al. [10]

FoM	**Observability**	**Stationarity**	**Driver Influence**	**Actuarial Relevance**
Acceleration	Medium	Low	High	Medium
Braking	Medium	Low	Medium	High
Speeding (Absolute)	High	High	High	Medium
Speeding (Relative)	Medium	High	High	High
Swerving	Low	Low	Medium	Low
Cornering	Medium	Medium	High	Medium
Location	High	High	Low	Medium
Smoothness	High	High	Medium	Low
Elapsed Time	High	High	Low	Low
Elapsed Distance	High	High	Low	High
Time of Day	High	High	Low	High

It is hypothesised that events with a low or medium stationarity in Table 3.1 would be more accurately classified using MEMS sensors when compared to using GPS data. Händel et al. [10] mentions the low observability and stationarity of some events as the reason for their low actuarial relevance. If these events' FoMs (Swerving, Harsh Braking, Harsh Acceleration and Harsh Cornering) can be accurately quantified it would add significant credibility to any vehicle monitoring system. In addition to these manoeuvres an attempt is made in Section 3.3 to extract GPS-specific data such as speeding, location (relative to known GPS points) and elapsed distance from the

MEMS sensor data. Section 3.3 discusses, in detail, the comparison of GPS and MEMS sensor methods for detecting these manoeuvres.

Two additional events, not included in Table 3.1 by Händel et al., are deemed relevant and included in this thesis. It is common practise in many countries to implement traffic calming devices in the form of speed-bumps. These devices have vertical deflections designed to limit discomfort when traversing below recommended speeds [43]. Measuring vertical acceleration over these devices can give an indication of the speed relative to the recommended speed.
Rough road surfaces, caused by cracking, surface deformation, disintegration or surface defects can reduce the traction between a vehicle's tyres and the ground [44]. The identification of high speeds over rough road surfaces is therefore also attempted.
A U-turn is considered as a distinct event (as opposed to the same as a turn), because it holds a significantly higher inherent risk as opposed to a standard turn.
To summarise, the events that will be focused on in this work is the fully descriptive set formed by:

- Acceleration
- Braking
- Lane Changing
- Swerving
- Speed Bumps
- Turns
- U-turns
- Rough Road Surfaces
- Coasting
- Stationarity

3.3 Choice of sensors

This section describes a method for evaluating the performance of two sensor systems for smartphone-based vehicle monitoring– GPS and MEMS sensors – in terms of the usefulness of data that can be collected for detecting and classifying driving manoeuvres, as well as the convenience of use and the effect on battery life. Comparisons are done in terms of selected Figures of Merit (FoM) discussed in Section 3.2. The manoeuvre specific data set described in section 3.4.1 is used for the comparison. The convention used for x-,y- and z-axes is discussed in Section 4.3.4.

The use of GPS positioning, presents a few significant obstacles that are not experienced when using inertial sensors:

1. The sample rate of un-augmented smartphone-based GPS is at best about 1Hz, which is too slow to detect and classify certain manoeuvres accurately, compared to smartphone-based MEMS sensors that have update-rates in the order of 50-300Hz.

2. GPS has a high energy consumption. Due to the slow (50 bps) data throughput of the GPS system, GPS antennas are kept active for longer periods than high-speed GSM-antennas. This is contrary to the prevalent method employed for reducing smartphone battery drain, where high speed communications are used and the antennas and processor are deactivated periodically to reduce energy consumption. [45]. GPS's average power drain while sampling at 1Hz is close to 370mW, while the average power consumption of sampling three MEMS sensors (gyroscope, accelerometer, magnetometer) at 100Hz is closer to 60mW (refer to Table 3.2 on page 35).

3. Un-augmented GPS readings have inaccuracies. Smartphone-based GPS location data is accurate to between 5 and 18m, a figure that can deteriorate during overcast conditions [46]. This considerably reduces the usefulness of GPS data for characterising small driving manoeuvres.

4. Reception of GPS signal is often inconsistent. Tunnels, forests, high-buildings and incorrect placement within a vehicle can all cause a GPS device to lose its locational fix. In contrast, inertial sensors are entirely self-contained.

5. Initial acquisition of locational fix can take up to 12 minutes (cold start, no A-GPS). This time is drastically reduced by Assisted-GPS (A-GPS) technology, but A-GPS requires cellular data, which is not always available and can incur additional data charges and battery drain. In contrast, MEMS sensors can be sampled on-demand with minimal adverse effects.

3.3.1 Sensor selection criteria

A series of tests were devised to produce quantifiable comparisons between GPS and MEMS sensor based sensing of selected FoMs in addition to battery drain and sampling rate. The smartphone-based logging application discussed in Section 3.5.1 was used to record required data for each of the tests.

3.3.1.1 Acceleration

The acceleration FoM is defined by Händel et al. [10] as: "The number of rapid acceleration events and their harshness."

The positive x-axis value (a_x) of the MEMS accelerometer's reading indicates longitudinal acceleration. To obtain acceleration from the GPS data, however, the velocity measurement must be differentiated as shown in Equation 3.1. When this acceleration exceeds a certain threshold it can be counted as a harsh acceleration. The harshness of the acceleration can be quantified by the magnitude of the acceleration.

$$a(t) = \frac{v(t) - v(t - \Delta t)}{\Delta t} \tag{3.1}$$

Figure 3.2 shows the acceleration of the vehicle over a 25 second interval. It is clear that, despite the similarities in acceleration magnitude, the GPS cannot capture the higher frequency acceleration events. The filtered accelerometer data shows the higher peak accelerations, that the 1Hz differentiated GPS speed fails to detect. From the figure, the gear shifts are also clearly visible at 4, 10 and 18 seconds. The GPS data also has a delay with respect to the acceleration. Though the error is small over the long

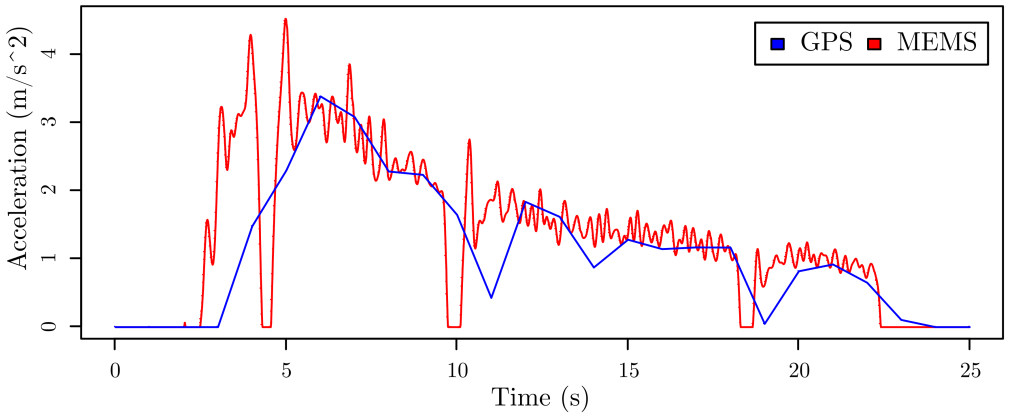

Figure 3.2: A plot of the vehicle's acceleration over a 25 second interval.

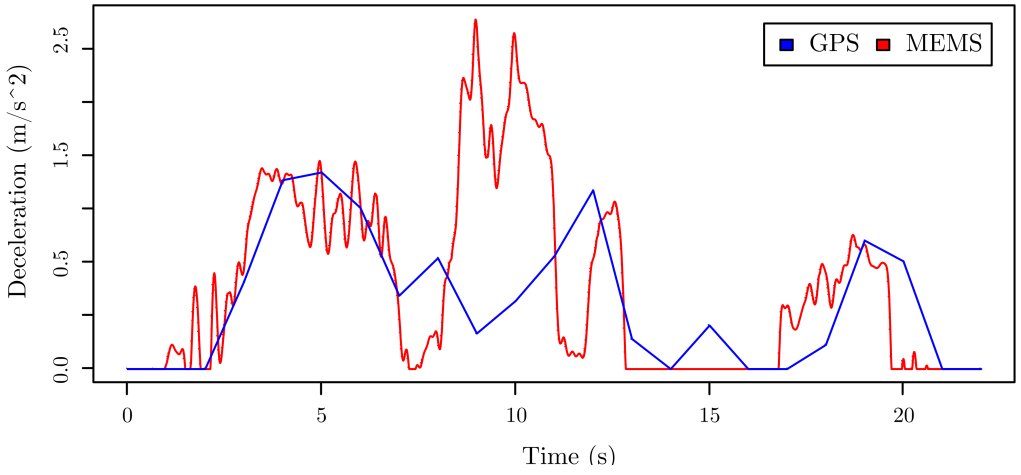

Figure 3.3: A plot of the vehicle's deceleration over a 22 second interval.

term, the detection of sharp accelerations and gear shifts are useful for Smartphone Based Vehicle Monitoring and Telematics and the MEMS sensors clearly perform much better.

3.3.1.2 Braking

A similar approach to the one used for the acceleration test (3.3.1.1) can be employed here, but the negative x-axis acceleration, $-a_x$ will be used and compared to the differentiated and negated GPS speed.

Figure 3.3 shows three minor vehicle deceleration events. Similar to the results with acceleration, the GPS cannot record the sharp accelerations as effectively as the MEMS sensors, though the average acceleration over the period is similar for both systems.

3.3.1.3 Speeding (absolute)

The smartphone-based GPS provides a location which is typically accurate to within about 18 meters [46]. However it also provides a scalar speed accurate to within $2m.s^{-1}$ calculated with the aid of the Doppler shift in received signals [47]. To estimate the speed of the vehicle using inertial sensors, the following three methods can be used:

Integration Estimating the speed of a vehicle accelerating or decelerating in a straight line is done by using Equation 3.2.

$$v_{est}(t) = v(t - \Delta t) + a_x(t) \times \Delta t \tag{3.2}$$

Turn speed estimation When the vehicle is in a turn the speed can be estimated by using the formula in Equation 3.3. Therefore:

$$v_{est}(t) = a_y(t)/\omega_z(t) \tag{3.3}$$

If the vertical axis acceleration (a_z) is below a predefined threshold for a number of consecutive samples, indicating a lack of road generated vibrations, the vehicle can be assumed to be stationary ($v_{est} = 0$).

Road irregularity autocorrelation A final method for determining speed using inertial sensors requires a single GPS velocity reading per vehicle as calibration method. As demonstrated by the SenSpeed system developed by Hun et al., a vehicle passing over a road irregularity, such as a speed bump or reflective road studs creates a disturbance in the z-axis acceleration for each wheel passing over the irregularity [48]. If the time between these disturbances (Δt_w) is calculated and the Wheel Base (WB) of the vehicle is known, the speed can be calculated using Equation 3.4.

$$v(t) = \frac{WB}{\Delta t_w} \tag{3.4}$$

The discrete autocorrelation of the z-axis acceleration data can yield the time between disturbances, as the z-axis acceleration signature is similar for both wheels.

The wheelbase can be requested from the user, but this is inconvenient as the wheelbase of their vehicle is not common knowledge for most users. Instead, it is proposed that the system be calibrated by using the GPS speed until an accurate wheelbase length can be estimated using the reverse of Equation 3.4.

If the detection of speeding is deemed a necessity and the accuracy of the speed estimated using the inertial sensors is insufficient, this speed can be used as a trigger, to request a more accurate GPS-based speed when within a certain threshold of the maximum legal speed limit. This threshold can be determined by quantifying the typical accuracy of the inertial speed estimation.

Since speed calculated by GPS is accurate to within $2m.s^{-1}$, the GPS speed will be used as a benchmark to quantify the accuracy of the speed estimated with MEMS sensors. Figure 3.4 shows the speed estimated by MEMS sensors via accelerometer integration (3.3.1.3) and turn speed calculation (3.3.1.3). This sample has been chosen to illustrate the drift in estimation when there are no fixed reference speeds to work with. The large error in this sample is caused by a variation in road incline, conflicting

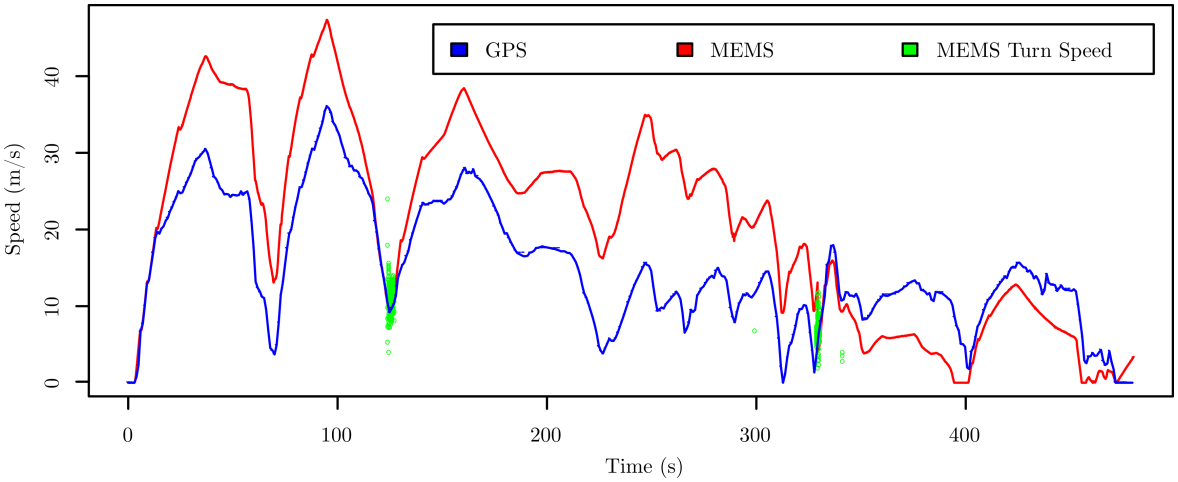

Figure 3.4: A plot of the speed of a vehicle, estimated via MEMS sensor integration (Red), MEMS sensor turn speed estimation (Green) and GPS Speed (Blue)

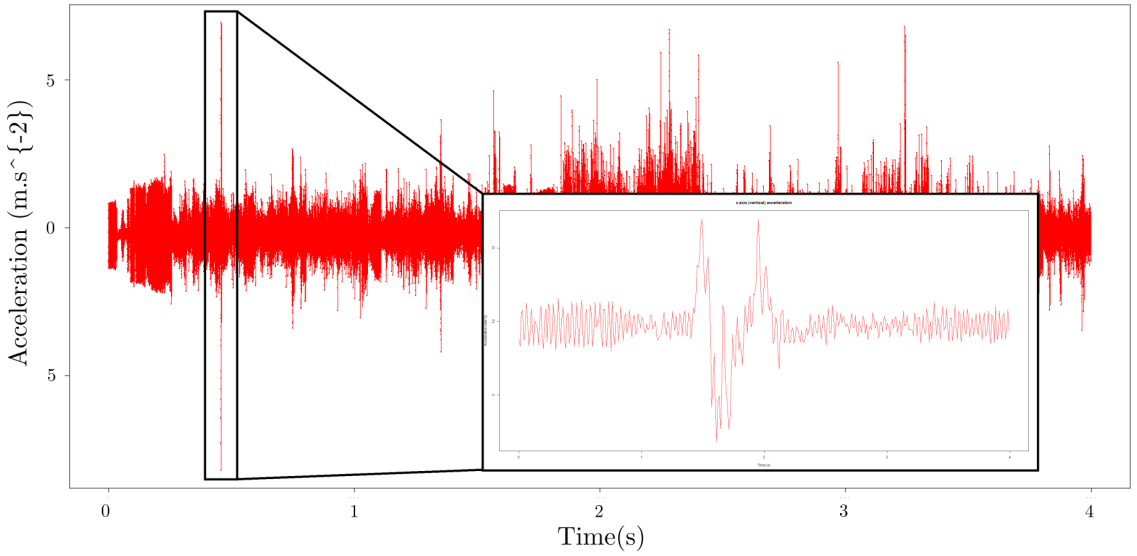

Figure 3.5: An extract from a plot of the z-axis (vertical) acceleration measurement versus time. The two impulses in the extract represent the front and rear wheels respectively impacting with a speed bump.

with the assumption that tests are carried out on a level surface. The slight incline change causes the accelerometer to measure a component of the earth's gravitational acceleration, thereby causing a drift in the speed estimation. The effect of gravity can be mitigated using the methods discussed in Section 4.7.1. The turn speed estimation was found to be accurate only for turns with a high yaw rate, where the integrated speed can then be corrected. Those corrections are shown in green in Figure 3.4. The other method for estimating speed through autocorrelation of the z-axis acceleration was found to perform well at low speeds of up to 40 km/h. Figure 3.5 shows a sharp z-axis acceleration caused by driving over a speed-bump at 20km/h. The two spikes where the front and rear wheels hit the bump are clearly visible. For this case, the autocorrelation of the extracted data had a local maximum at 490ms. The vehicle in consideration has a wheel base of 2.6m, which in turn gives an estimated speed of 19.1 km/h while the GPS speed at that point in time is 21.38 km/h.

3.3.1.4 Speeding (relative)

Relative speeding refers to determining whether a vehicle is speeding, relative to the applicable speed limit at its current location. The road on which the vehicle is driving must be identified and an external map database with updated speed limit for the applicable road must be consulted. This typically requires constant and fine GPS positioning.

Another method proposed to determine the local speed limit is a combination of the speed estimation done in Section 3.3.1.3, a digital speed limit database (an example of a free community-driven one is Wikispeedia) and Geo-fencing.

Geo-fencing is a method of setting a digital perimeter of a specified radius around a specified point and monitoring whether a device is within or outside of the perimeter. This method of providing location awareness is useful as it allows the phone to only use energy consuming GPS positioning when the accuracy is required. When accuracy is of little importance, more energy-efficient positioning such as, cellular tower triangulation or WiFi positioning can be used.

In an embodiment of this method, an application can be designed to, initially, obtain a fine GPS-location fix and query the speed limit database to find the current speed limit and the closest different speed limit. A Geo-fence can now be defined with a radius equal to the distance to the nearest change in speed limit. Once the perimeter is crossed, this process can be repeated, thereby insuring the minimum amount of GPS usage.

The inertial navigation system (INS) discussed in Section 3.3.1.7 could also be used instead of Andriod's built in Geo-fencing API. The testing of this method is beyond the scope of this thesis. The legal aspects of this form of reckless driving detection requires further study and anonymous participation data for testing.

3.3.1.5 Swerving

Swerving is generally defined as "causing to change direction abruptly". Swerving typically happens when trying to avoid an object in the path of the vehicle, performing abrupt lane changes or making coarse course corrections. These maneuvers are exceedingly important to detect and classify as they are effective indicators of drunk-, fatigued-, distracted- and aggressive driving [23, 49, 7].

Using inertial sensors, swerving is manually identifiable as a brief spike in either lateral acceleration (a_y) or yaw rate(ω_z), the magnitude of the lateral acceleration is an effective indicator of the harshness of the swerve as a higher lateral acceleration equates to a higher probability of the vehicle skidding or rolling.

Identifying a swerve via GPS is more challenging. The yaw rate as shown in Equation 3.5 and the lateral acceleration of Equation 3.6 can be estimated from GPS parameters. As swerving is often a very brief event and the change of direction is typically corrected almost instantly to coincide with the original heading, GPS, due to its 1Hz update rate, will seldom be able to identify a brief swerve event, much less the harshness of it.

$$\tilde{g}_z(t) \approx \frac{(h(t) - h(t-1))}{\Delta t} \tag{3.5}$$

With $h(t)$ being the heading obtained from GPS at time t.

$$\tilde{a}_y(t) \approx \frac{v(t) \times \sin(h(t) - h(t-1))}{\Delta t} \tag{3.6}$$

As shown by Figure 3.6, the MEMS sensors excel at detecting the brief lateral accelerations and rotations associated with swerves. GPS, however lacks the update rate necessary for detecting events shorter than two seconds.

3.3.1.6 Cornering

The definition used for cornering will, in this thesis, differ from the one given by Händel et al., who describes the cornering FoM as: "The number of events when turning at too high speed and their harshness". As shown by Zeeman et al. and Wahlström et al. [50, 34], the two main dangers of harsh cornering are tyre slip and roll-over, both of which are indicated by a large lateral acceleration.

Wahlström et al. [34] proposes a method for extrapolating turn harshness from GPS data. Their method involves a Kalman filter algorithm applied to the parameters of the GPS to continuously update an estimate of the ratio between the normal and horizontal force on the vehicle's tyres. This ratio determines the threshold at which the vehicle will slip. Despite the large amount of processing done on the GPS data, the amount of missed detections and false alarms summed to approximately 40% of the total number of cornering events.

As with swerving, cornering events and their harshness can be visually identified from inertial sensor readings. An extended increase in lateral acceleration (a_y) and yaw rate (ω_z) indicates a cornering event, and the magnitude of the lateral acceleration (a_y) is proportional to the harshness of the cornering event.

The lateral acceleration can be recorded directly by the MEMS sensors during cornering as shown in Figure 3.6, avoiding the necessity for calculating lateral acceleration. The implementation of an algorithm for acquiring accurate lateral acceleration data from GPS data is considered beyond the scope of this thesis. Suffice it to say that, identifying and rating the harshness of cornering events is significantly easier and more computationally efficient using MEMS sensors than GPS.

3.3.1.7 Location

Location data is useful in a smartphone based vehicle telematics system for three reasons. Firstly, certain geographical areas can be flagged for a higher rate of accidents and thereby increasing the risk of driving there, secondly, the location of the vehicle can be useful to transmit for telematics purposes and remote monitoring and finally, the location of an event on a map is a useful way of providing post-driving feedback to a driver.

A simple and computationally efficient inertial navigation algorithm is developed below to obtain the 2-dimensional Cartesian coordinate position, heading and speed of the vehicle from the Accelerometer, Magnetometer and Gyroscope data.

The inertial navigation algorithm proposed here is based on the following basic methods, and the algorithms are deliberately chosen to be as simple as possible for the purpose of computational efficiency. Heading can be estimated by Equation 3.7 and, if the magnetometer has been deemed calibrated, the proprietary Android rotation

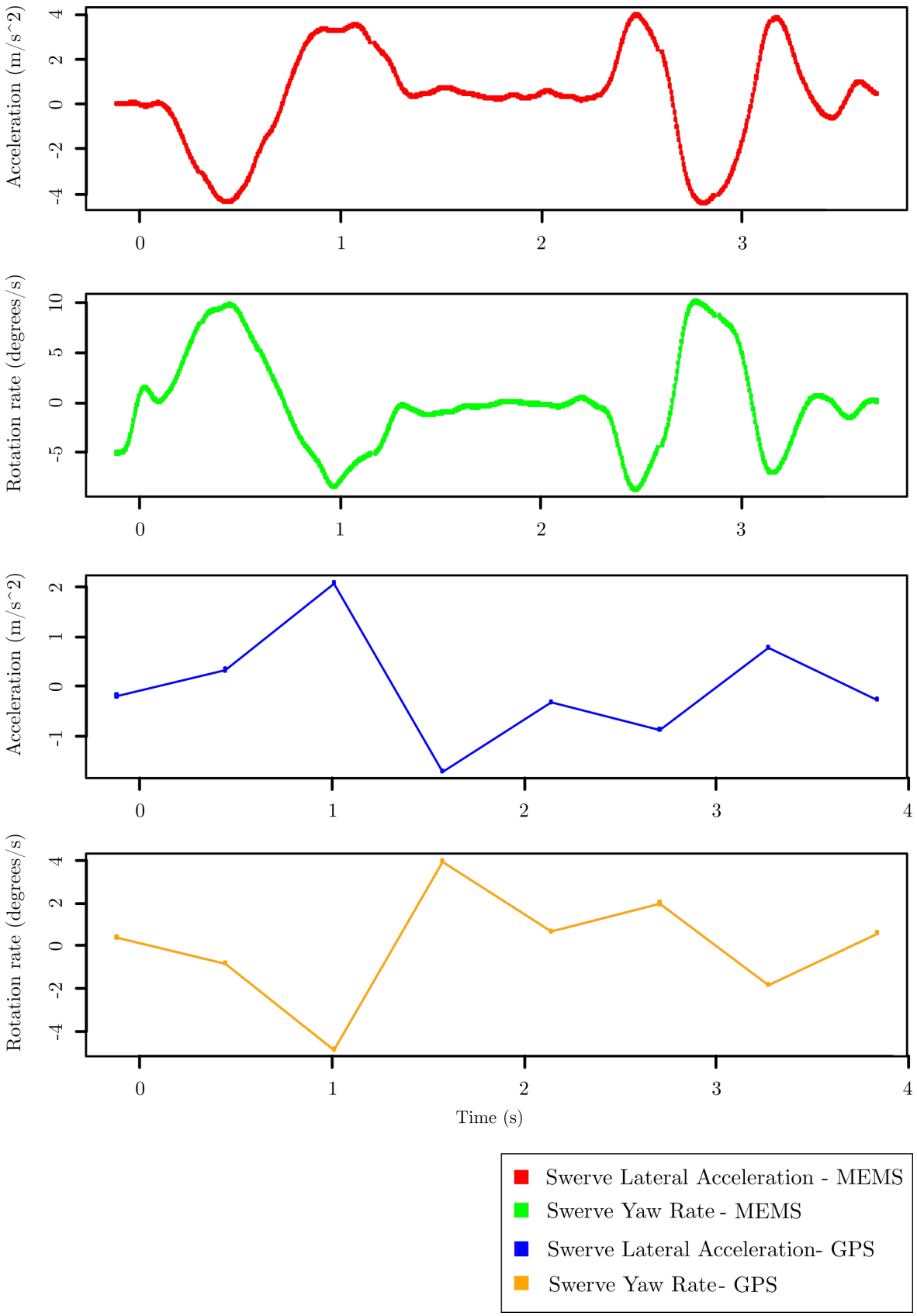

Figure 3.6: A series of plots showing the lateral acceleration and yaw rate measured by the MEMS sensors and estimated by the GPS over a seven second period containing a brief lane change and a swerve around an object, respectively.

vector synthetic sensor can be used to calculate the heading.

$$h_{est}(t) = h(t - \Delta t) + \omega_z(t) \times \Delta t \tag{3.7}$$

Where h_{est} is the estimated heading, Δt is the sampling period, $q_(x, y, z)$ are the complex parts of the quaternion corresponding to the Cartesian axes and qw is the scalar component of the quaternion as explained in section 4.3.4.

$$h_{est}(t) = atan2(2 \times q_y \times q_w - 2 \times q_x \times q_z, 1 - 2 \times q_y^2 - 2 \times q_z^2) \tag{3.8}$$

To calculate the coordinate position from MEMS sensor data, an initial point must be defined as the start of the event, the origin (0,0) is defined as the starting point. Given the starting coordinate $C(0) = (C_la(0), C_lo(0)) = (0, 0)$ The next coordinate can be calculated as shown in equations 3.9 and 3.10:

$$C_{la(est)}(t) = C_{la(est)}(t - \Delta t) + v_{est}(t) \times \Delta t \times \cos(h_{est}(t)) \tag{3.9}$$

$$C_{lo(est)}(t) = C_{lo(est)}(t - \Delta t) + v_{est}(t) \times \Delta t \times \sin(h_{est}(t)) \tag{3.10}$$

The coordinates estimated by equations 3.9 and 3.10 are, in contrast to the GPS measurements, in meters, relative to the start of the event and not in decimal degrees relative to the Prime Meridian at the equator.

For the sake of comparison, the initial GPS coordinates $C(0)$ are subtracted from each subsequent GPS coordinate and multiplied by an estimation of the earth's radius at that position ($6370 \times 10^3 m$ at $-33\,\mathrm{deg}\,latitude$), thereby recording the coordinates in meters relative to the origin or C(0).

The comparison of the inertial navigation algorithm with the GPS data showed that, as was expected, integrated noise of the accelerometer and gyroscope measurements led to large errors over time. However, for single manoeuvres or events, lasting less than a minute, the system did prove accurate at mapping the coordinates as long as the initial velocity was accurate. Figure 3.7 shows the drift in speed and coordinate position calculated by inertial navigation algorithm.

3.3.2 Additional technical criteria

3.3.2.1 Battery Drain

To verify the statistics on battery drain, the time it takes for the application to drain a full battery while logging GPS data, logging MEMS sensor data and idle is recorded. All logging is done at maximum sample rate. The screen is kept on for monitoring purposes. The battery drain (in mA) can then be calculated as shown in Equation 3.11.

$$I_{avg} = \frac{Capacity(mAh)}{DrainTime(h)} \tag{3.11}$$

Table 3.2 summarises the battery drain test statistics for the Samsung Galaxy S4 with a 2600mAh battery:

The relative current drawn by GPS is therefore more than 5 times that of the MEMS sensors.

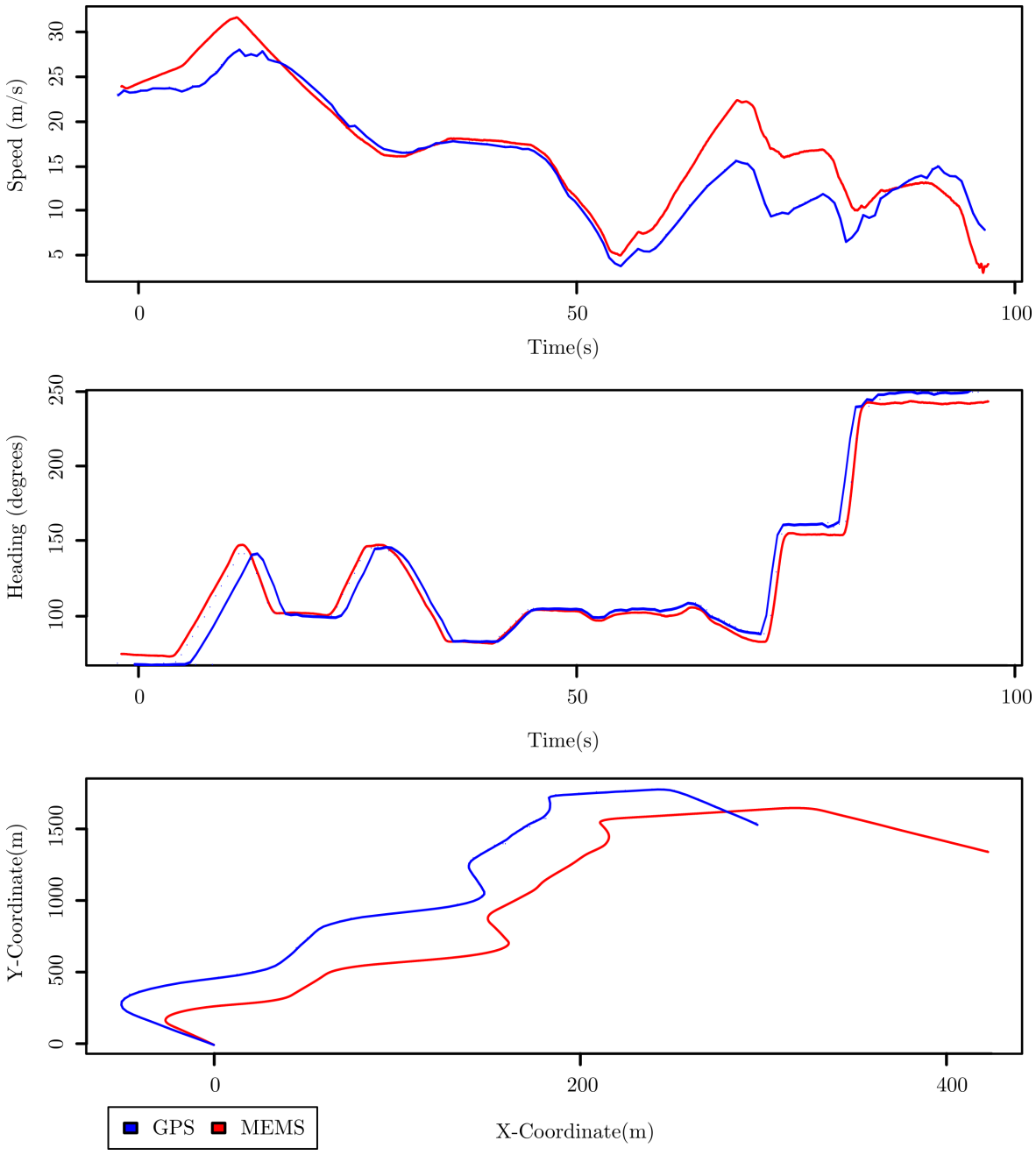

Figure 3.7: Three plots comparing the speed, heading and coordinates estimated by the MEMS sensors and measured by the GPS.

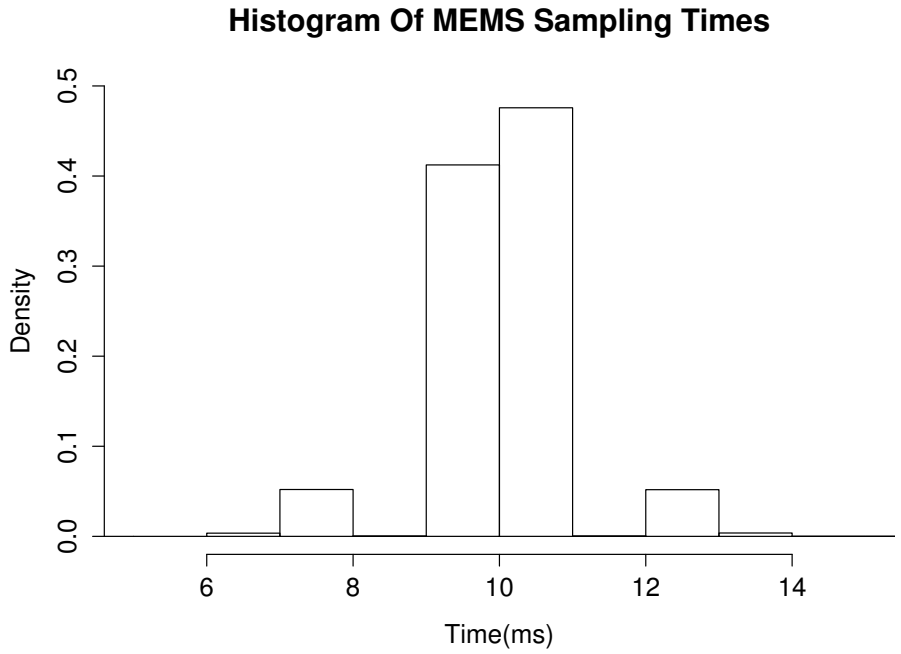

Figure 3.8: A histogram of the MEMS sensors' sampling period.

3.3.2.2 Sampling Rate

Statistics on the sample rate and variance in sample rate of the various sensors are obtained from logged driving data.

The highest possible sample rate of the GPS is 1Hz ($T_s = 1s$), while the MEMS sensors are around 100Hz ($T_s = 0.01s$). This is a significant advantage for the MEMS sensors. The consistency of the sample rate is also important however, and here the MEMS sensors have a slight variation in sampling period. Figure 3.8 shows a histogram of the MEMS sensor sampling periods obtained by calculating the time between all consecutive samples.

The GPS is within 10% of its correct sampling period 99.81% of the time, while the MEMS sensors are within 10% of their correct sampling period 88.81% of the time.

3.3.3 Conclusion

It is clear from these tests, that MEMS sensors outperform GPS in various aspects of detecting the manoeuvres of relevance in this thesis including the detection of swerving as well as sudden accelerations and decelerations. As shown in the summarising Table 3.3 however, there are other areas where GPS is still indispensable, such as providing locational data. Due mostly to its lower power consumption and its independence of

Table 3.2: Results of tests on battery drain for three MEMS sensors versus GPS.

	Idle	GPS	MEMS
Drain time (hh:mm:ss)	04:56:25	04:08:39	04:47:26
Current draw (mA)	526	627	542
Relative current draw (mA)	-	101.08	16.44

Table 3.3: A summary of the comparison between MEMS sensor and GPS methods.

Test:	GPS:	MEMS:
Acceleration	Misses brief accelerations	✓
Braking	Misses brief decelerations	✓
Speeding	✓	Inaccurate without reference points (turns, bumps)
Swerving	✗	✓
Cornering	Lateral Acceleration requires complex algorithms	✓
Location	✓	Only brief events
Battery Drain	101.08 mA	16.44 mA
Sample Rate	1 Hz	100 Hz

unreliable external communications it is the view of the author that MEMS sensors holds the advantage for use in this thesis.

3.4 Data capture scenarios

A prerequisite resource for attempting this project was a data set that could be used to analyse typical smartphone-sensor data, train and test proposed algorithms and evaluate findings relative to ground truths. Three methods were used to collect data, each with a specific purpose in mind, all collections were done using one or both of the two applications developed (Logging app and flagging app). Test one was the manoeuvre specific data collection, used for the comparison of GPS and MEMS sensor in Section 3.3 and the improvement of the logging app, the second test had a fixed and a moving logging application in the vehicle to provide a test setup for the reorientation algorithms. The final test took place on a specially chosen route with various drivers and the addition of the flagging application to create a dataset with annotated (flagged) manoeuvres.

3.4.1 Manoeuvre specific data collection

This simple testing scenario involved performing each of the manoeuvres listed in Section 3.2 with a vehicle while recording the data. The data was segmented so that each recorded CSV file represents one of the manoeuvres.

3.4.2 Varying orientation test setup

An important objective of this thesis (objective 5) is being able to determine the orientation of the phone relative to the vehicle while the application is running and without any user interaction. A dataset is needed that allows us to test if the algorithms that calculate and perform the rotation from the sensor to vehicle axes are effective. The test must provide data from a phone in an unknown orientation within the vehicle as well as data that has been recorded from a position exactly aligned with the vehicle axes (ground-truth data).

A test is set up where two logging applications are used within a moving vehicle. One application is fixed to a known orientation that coincides perfectly with the vehicle

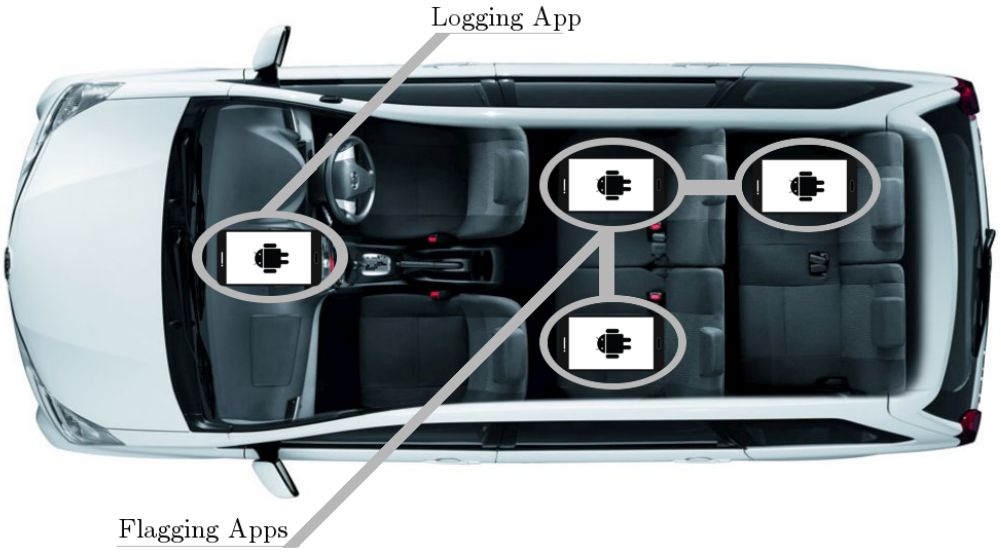

Figure 3.9: A diagram illustrating the positions of the logging app and the passengers with the flagging app within the Toyota Avanza used for testing.

Table 3.4: Table summarising the number of manoeuvres recorded per trip.

Maneuver	Speed < 40km/h	40km/h<Speed< 90km/h	Ascent	Descent
Turn<45	3	>10	3	3
45<Turn<90	>10	6	3	3
90<Turn	2	2	1	1
Acceleration	>10	4	4	1
Brake	>10	>10	1	8
Lane Change	1	4	1	1
Swerve	± 3			
Road Surface	3			
Speed Bump	9			
U-Turn	2			

axes. Another is intermittently moved within the vehicle to different orientations. This process is continued during a variety of vehicle manoeuvres.

3.4.3 Annotated data aggregation setup

The data aggregation process involves one phone running the logging application and one phone for each person in the vehicle (excluding the driver) running the flagging application. The logging application needs no user interaction during operation aside from initially starting it. It is placed in a standard window-mounted smartphone cradle. The flaggers all sit in the rear seats of the vehicle to minimise distractions for the driver and to ensure that all users have the same experience of the driving. The vehicle used for the flagging tests was a 7 seat Toyota Avanza SX with a 1.5l petrol engine as shown in Figure 3.9. Tests were done on a predetermined route, chosen to maximise the number- and variety of driving events at different speeds and inclines as summarised in Table 3.4.

Each test run was executed with the help of 17 volunteers and the author. Each test involved one driver and 2-4 flaggers. The flaggers used the flagging application on

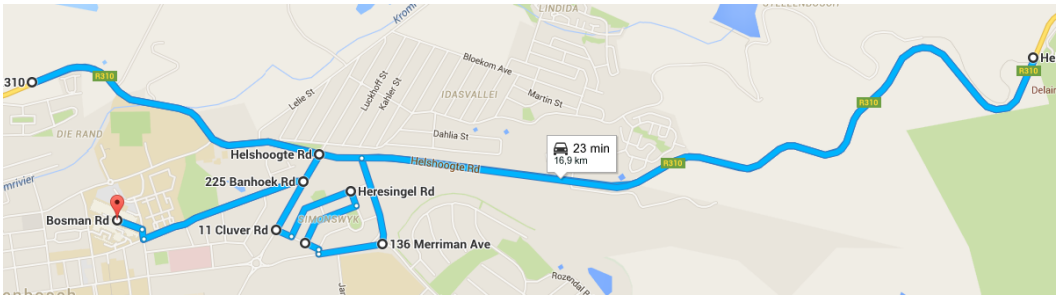

Figure 3.10: The route followed for data capture.

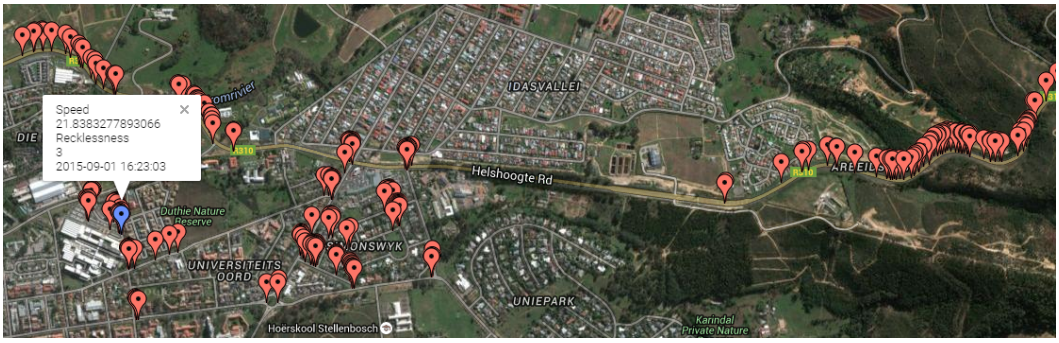

Figure 3.11: A map showing the braking events for 8 runs of the route. An interactive HTML map of all classified data was created for visualising the data in Google Maps.

their smartphones to flag, rate and describe manoeuvres. A dash camera is used to record visual footage of the driving in order to facilitate post-test review of inconsistent flags. The complete route takes on average 28 minutes to complete, after which the driver is rotated.

The testing procedure had the following chronological steps:

1. Before the volunteers arrive, the phone that is used to run the logging application is placed in the window-mounted smartphone cradle in the middle of the front windscreen and plugged into a USB-charger. A dash camera is mounted below the smartphone cradle facing the road. The phones which are to run the flagging applications are fully charged.

2. The vehicle is then driven to a pre-determined location where the road was measured to be level. While stationary at the level location, the logging app is activated and set to record and display the accelerometer values. The cradle is rotated so the accelerometer reading coincides as closely as possible with the smartphone's positive y-axis (see Section 4.3.4 for axes conventions), thus aligning the sensor's y-axis with the z-axis of the vehicle. The vehicle is then accelerated in a straight line and brought to a halt. The recorded data is then stored in a file showing the session time and date. Using this calibration data the exact orientation of the sensor relative to the vehicle can be calculated and the data can be rotated accordingly.

3. The group of volunteers are then picked up at the route's starting point. The following is explained to them:

    - How the flagging applications works.

- What the different manoeuvres are.
- That their rating of the vehicle should be based on their perception of recklessness where 5/5 would be classified as a reckless event.
- In order to drive they need to have their drivers licences with them.
- All traffic laws are to be strictly obeyed.
- Data recorded will be anonymised.
- The route is explained in detail including the speed limits.

4. The setup procedures for the flagging and logging apps are completed.
5. A brief (approximately 5 minute) test drive is done in order for the volunteers to familiarise themselves with the flagging application's operation.
6. All files recorded during the test-run are discarded and the applications are re-initialised and restarted.
7. The author drives through the route while the volunteers flag their manoeuvres.
8. The recorded data from all applications is stored and the applications are re-initialised and restarted.
9. The next driver drives the route.
10. When all drivers have driven the route, data is retrieved from all the phone memory cards and backup copies are made.

In total, more than 20 hours of driving data was aggregated, containing 5390 events correlated to more than 10 million time samples from 7 sensors(Accelerometer, Gyroscope, Magnetometer, Gravity, Linear Accelerometer, Rotation Vector, GPS).

3.5 Data capture tools

The following sections discuss the applications used and the test set-ups.

3.5.1 Logging application

The logging application (screenshot shown in Figure 3.13) was designed to record smartphone sensor data on demand and give an accurate representation of the data that would be received from a generic Android phone with the prerequisite sensors.

3.5.1.1 Understanding Android sensing

There are nine hardware sensor types that can be sampled using the Android API: accelerometer, gyroscope, magnetic field, pressure, proximity and temperature only the first three hardware sensors are directly useful for reckless driving detection and are recorded. In addition to the hardware sensors, the Android API provides a set of "software sensors", which are essentially processed combinations of data from the hardware sensors [41]. For example: the *rotation vector* is a "software sensor" combing the data from the accelerometer, gyroscope and magnetometer to estimate the orientation of the device relative to the earth's axes.

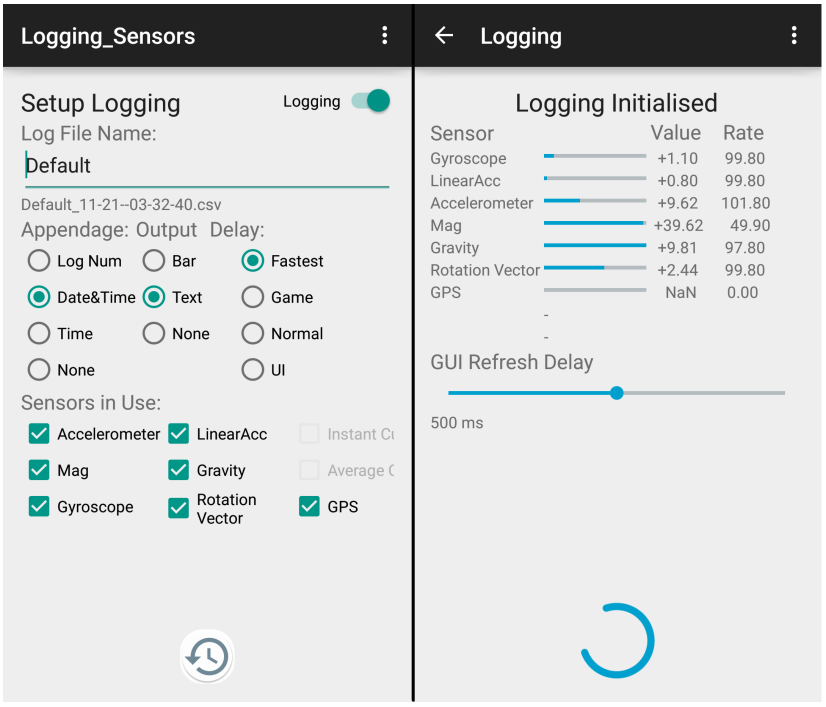

Figure 3.12: A screen shot of the setup screen (left) and one of the active views(right) of the logging application.

Android API 21 and later allows the app to obtain the phone's instantaneous and average "current draw", which can be used to quantify the efficiency of specific sensors. Finally, using Android's location manager class, GPS data specifying the phone's latitude, longitude, speed and heading can be acquired.

For sampling these sensors, Android provides 2 different callback functions: *onLocationChanged* for the GPS data and onSensorChanged for the rest. These callbacks are responded to concisely by checking the type of data received, converting it to a Comma Separated Value (CSV) string accordingly and sending it to a separate file writing thread.

3.5.1.2 Application functionality

The application consists of two activities: setup_logging and logging. The first allows the user to select various options w.r.t. the GUI, the storage of the data as well as the type and format of the data to be logged. In order to correlate the data recorded using the app with other collected data (i.e. flags, video) the first activity also ensures that the device-time is synchronised from a local time server before the logging activity is started. The logging activity, as shown on the right of Figure 3.12, can represent the data as text, bars or the user can choose to have no data displayed (useful in case of limited processing capacity). The GUI update speed can also be adjusted via a slider.

The application is written to handle Android life-cycle changes safely by ending any recording in *onPause*, de-registering sensor listeners and finishing the writing of recorded data to the relevant file before returning to the *setup_logging* activity.

3.5.1.3 Testing application

Elementary testing of the data produced by the application was done by recording all sensors during one of each manoeuvre specified in Section 3.2. The resulting data was imported and processed with the minimal processing methods specified in Section 4.4, visualised using the web application developed in Section 3.7, and compared to expected results. The accelerometer accuracy was tested by comparing the magnitude of gravity in each of the three sensor axes to the known value of $9.796 m.s^{-1}$. The gyroscope bias was measured while the phone was stationary. Due to the lack of a 3-axis turntable, the gyroscope accuracy was checked by rotating once around each axis, integrating the resulting data and comparing with the expected value of 2*Pi. All these tests were done using 3 Android smartphones: The Galaxy S3 mini, Galaxy S4 and the Nexus 5. The results showed a considerable and varying gyroscope bias, which would cause problems if not mitigated when using inertial navigation algorithms. The accelerometer had between 1% and 8% scaling error on the tested phones. But the visual data showed that the data produced was of an adequate accuracy for the intended purpose.

3.5.1.4 Data format

All data is stored on the smartphone's external memory card in a comma separated value format similar to what is shown below:

```
Samplenumber,Timestamp,SystemTime,Acc.1,....,GPS4,Settings:...
0,11407032779914,1441271841043,0.254,...,23.4
```

A challenging aspect of working with the Android sensor APIs is that the timestamp accompanying any sampled data is not of a specified format, but varies according to the on-board MEMS sensor chip manufacturer. The format of the timestamp must therefore be verified and adjusted by an offset or scale-factor where necessary. This process is discussed in detail in Section 4.4.2.

Table 3.5 shows a summary of the data collected using the logging app if the sensor is aligned with the vehicle axes.

3.5.2 Flagging application

The purpose of the flagging application (screenshot shown in Figure 3.13) is to gather inputs from multiple passengers in a vehicle about the manoeuvres the driver is performing with the vehicle. The data is limited to the type of event and a perceived recklessness rating for the specified event. The flagging data is used to annotate the data from the logging app, therefore one must be able to accurately link the two datasets temporally.

3.5.2.1 Application functionality

The application consists of three main activities: Setup, Manoeuvres and Flagging. The setup activity provides a logical, easy to use checklist of settings that need to be specified before testing can begin. The user is guided through the process of synchronising the time from a local time server, loading any required manoeuvres in the "Maneuvers" activity, specifying their device ID for the test, creating the file (with correct naming protocol) and optionally verifying device cloud access. Only after all

Table 3.5: Sensor data received from Android API after being rotated to the vehicle axes.

MEMS Sensors			
Accelerometer: ($m.s^{-2}$)	Longitudinal acceleration (a_x)	Lateral acceleration (a_y)	Vertical acceleration including gravity (a_z)
Gyroscope: ($Rad.s^{-1}$)	Roll rate(ω_x)	Pitch rate (ω_y)	Yaw rate(ω_z)
Magnetometer: (μT)	x-axis magnetic field strength (m_x)	y-axis magnetic field strength (m_y)	z-axis magnetic field strength (m_z)
Linear Acceleration (Synthetic sensor): ($m.s^{-2}$)	Longitudinal acceleration (lin_x)	Lateral acceleration (lin_y)	Vertical acceleration excluding gravity (lin_z)
Gravity (Synthetic sensor): ($m.s^{-2}$)	Component of gravity in x-axis ($grav_x$)	Component of gravity in y-axis ($grav_y$)	Vertical Component of gravity in z-axis ($grav_z$)
Rotation Vector: (Synthetic sensor).			
x-axis Rotation component q_x ($x \times sin(\theta)$)	y-axis Rotation component q_y ($y \times sin(\theta)$)	z-axis Rotation component q_z($z \times sin(\theta)$)	Scalar component q_w($cos(\theta)$)
Global Positioning System			
Speed(v in $m.s^{-1}$)	Heading (h in deg)	Latitude(C_{la} in deg)	Longitude(C_{lo} in deg)

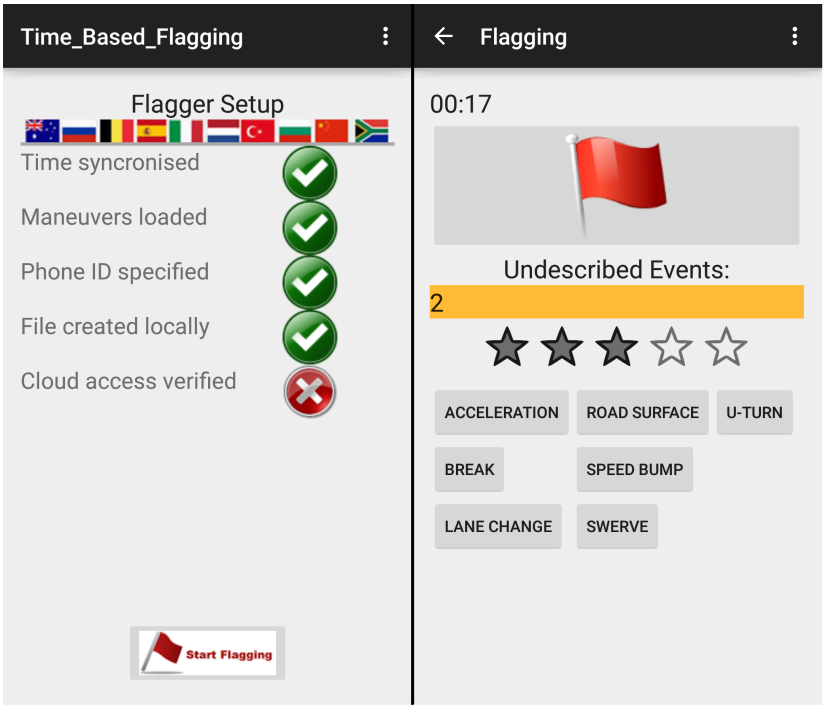

Figure 3.13: A screen shot of the setup screen (left) and one of the active views(right) of the flagging application.

settings are correctly specified does the button appear allowing the user to start flagging. In the flagging activity, a flagging button can be pressed to indicate the start of an event, and a rating and description can be added to the flag. The output of the app is a comma separated value file containing the flags in the form of a *Timestamp*, *Rating* and *Event*. The timestamp value can then be used to accurately synchronise the data in this dataset to that of the logging application. Figure 3.11 shows an example of how the flags collected from the flagging app can be used.

3.5.2.2 User interface design

Designing the flagging app so users could learn to use it intuitively within minutes and focus on the events themselves presented a challenge. When driving in urban areas, driving events tend to occur in close succession with less active sections in between. The application should facilitate the rapid flagging of events during active periods followed by the event descriptions during less active periods. The "Flagging" activity features a large button with clear visual feedback for pressing (the flag also changes colour) as the user needs to be able to accurately flag the start of each event. Flagging an event causes the timestamp at which the flag button is pressed to be pushed onto a FIFO queue. A rating can then be given to the event using an Android rating bar and the rating along with the event description is stored when the relevant event type is selected. The number of undescribed events in the queue is always shown so the user can see when undescribed events are piling up too high. The rating and event types are stored in a data structure along with the timestamp retrieved from the head of the timestamp-queue.

3.5.2.3 Testing application functionality

To test the app functionality, the app was loaded onto three smartphones and the logging app was installed on another. Three maneuvers were programmed onto the flagging app: "left", "right" and "shake". The users were then instructed to perform the rest of the setup on the flagging app. The logging app was set to record acceleration and was held by another person in a fixed orientation. The logging app was now moved periodically up, down or it was shaken while the flagging app users flagged the start of the movement gave it a rating i.t.o. intensity. The logging and flagging data were manually reviewed and showed excellent correlation i.t.o. the start, magnitude and type of movement.

3.6 Data storage

Due to the data-driven nature of the project it was essential to facilitate quick and convenient access to all collected data. The data that was aggregated has varying dimensions for each sample and the main unifying factor is the time at which each sample was acquired. A storage solution was chosen to enable the acquisition of the data mainly by searching the indexed timestamps of all the data.

MongoDB is an open-source Document-oriented database in a NoSQL format. Each sample of data acquired during the course of this thesis was stored as a single document consisting of multiple key:value pairs including, in all cases, a timestamp field. The timestamp field was indexed to allow speedy search and aggregation operations.

Figure 3.14: A screen shot of the slider used to select the time boundaries for the data to use.

NoSQL was chosen over a conventional SQL database due to the varying structure of the acquired data. MongoDB is also scalable to larger systems enabling convenient future expansion of the project.

3.7 Data visualisation web app

Raw data from accelerometers, gyroscopes and magnetometers as well as orientation or coordinate data tend to be quite abstract and difficult to fully comprehend from raw numbers. To extract as much information as possible from the data and to perform periodical sanity checks, the data has to be plotted or otherwise visualised often. A user interface was therefore developed to streamline interaction with the data. The R statistical computing language was used for data processing and R provides a framework for developing interactive web applications which interface directly with your standard R code. The tools that were regularly used for data processing and analysis were modified to receive parameters from the UI and display requested data visually. Figure 3.20 shows a screenshot of the running application.

3.7.1 Tools implemented

The default interface allows the user to one of the available collections in the Mongo database and to choose a recording session available in the specified collection. When a session is selected, a slider is updated, indicating the timeline of the session (labelled in seconds since session start) as shown in Figure 3.14.

The two slider selectors allow the user to select the timeframe of interest and the rest of the GUI is updated to reflect only the data within the specified timeframe. The following tools were added to the R Shiny web app.

3.7.1.1 MEMS sensor plots

Pre-processed gyroscope and accelerometer data can by plotted with user adjustable y-axis scaling and the option to add a LPF with the specified number of coefficients and cut off frequency. The original, unprocessed data can also be plotted with the same options. This facilitates a quick visual comparison showing the effects of pre-processing. Figure 3.15 shows a screenshot of these plots.

3.7.1.2 State

The "Training" data, consisting of the state that the vehicle was in during each time-interval (i.e. idle, accelerating, turning, etc.).These states are calculated from GPS and flagging data for the selected timeframe as discussed in Section 5.3. When the Hidden Markov Model tuning tool is active, the output states estimated by the HMM are shown on the same axes for visual comparison.

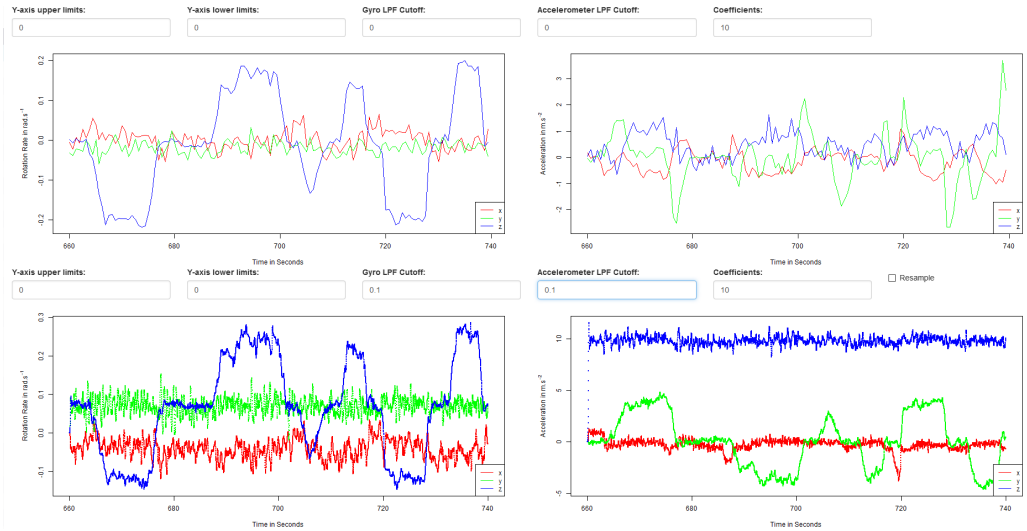

Figure 3.15: A screenshot of the MEMS sensor plots in the web app, a 10Hz LPF is applied to the unprocessed plots (bottom).

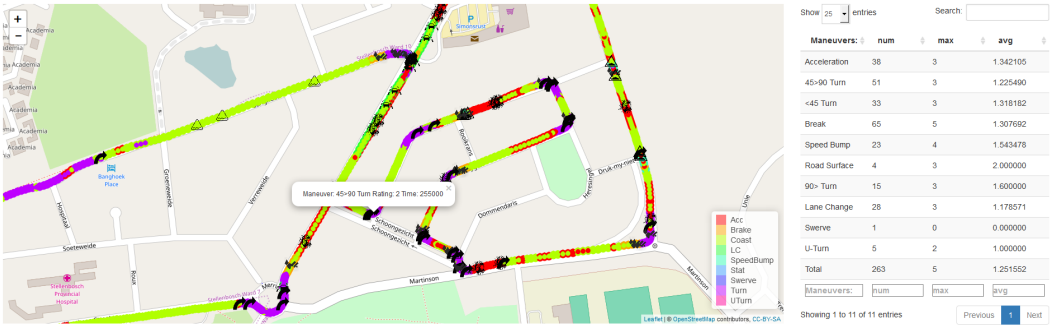

Figure 3.16: A screenshot of the interactive map and events summary table in the web app. The map shows the location of flags with applicable icons as well as 2Hz annotations and classifications of the data as concentric colour-coded outer and inner circles respectively.

3.7.1.3 Map, classifications and flags

All flags from the flagging tests as well as the route the vehicle followed (interpolated from GPS) are by default plotted on an interactive street map. clicking on any flag produces a brief description of the manoeuvre, its rating and the exact time it took place. If a database containing training labels and classification results is selected, these are shown as concentric, overlapping coloured dots. The large dot represents the training label assigned to the point and the smaller inner dot represents the result of the classification algorithm for that point. This provides an intuitive way of visualising and making sense of the results of the classification. An interactive table (right hand side of Figure 3.16) is produced showing a summary of each event type that has taken place within the specified time-frame as derived from the user flagging data.

3.7.1.4 GPS Speed and Heading

The heading and speed of the vehicle over time as given by GPS data can be plotted on the same axes as shown in Figure 3.17.

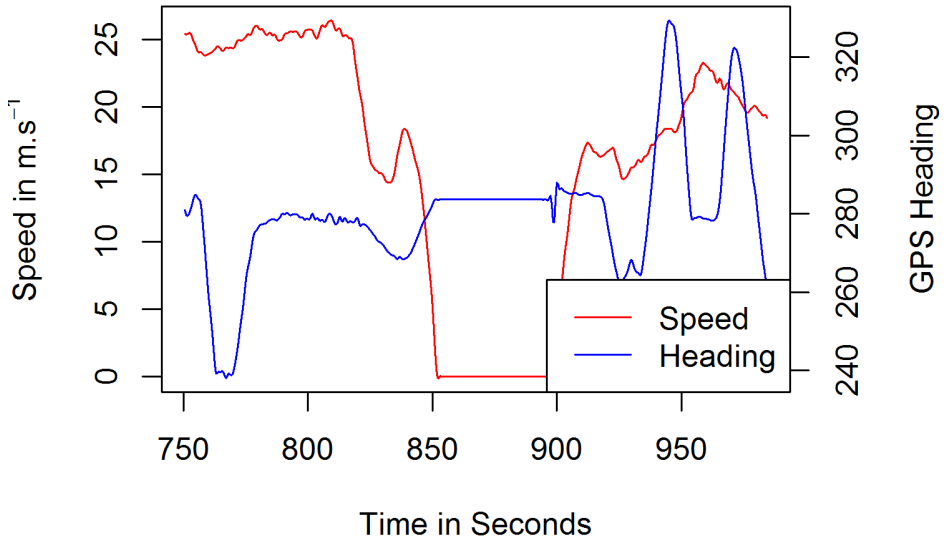

Figure 3.17: A plot of GPS speed and heading generated by the data visualisation web app, plots can be displayed on-screen, but the option is also provided to plot all outputs with a user-specified filename template.

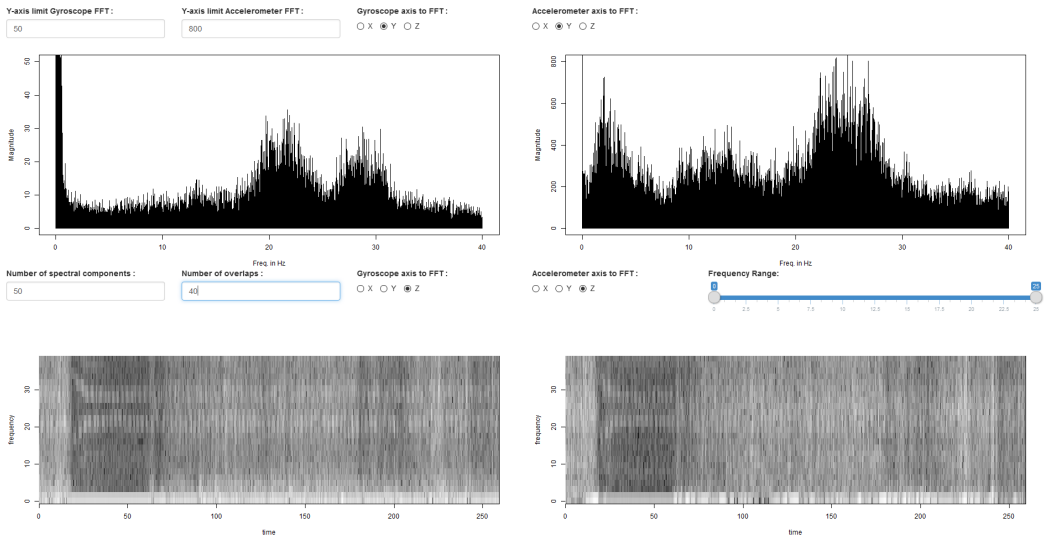

Figure 3.18: A screenshot of the FFT and spectrogram plots produced by the data visualisation web app.

3.7.1.5 Frequency information

Three distinct visualisations of the frequency-domain information in the accelerometer or gyroscope data can be produced. Any one of the gyroscope or accelerometer axes can be selected and the FFT of the selected data is plotted. A colour-scaled spectrogram with a selectable number of spectral components and hamming-window overlaps can be plotted to visualise frequency domain changes over time as shown in Figure 3.18.

The signal energy within a frequency range specified by a slider can be plotted w.r.t. time. The ratio of energy produced by points where the vehicle is stationary vs. where it is moving is also given as text. This is used to identify frequencies containing road noise or engine noise.

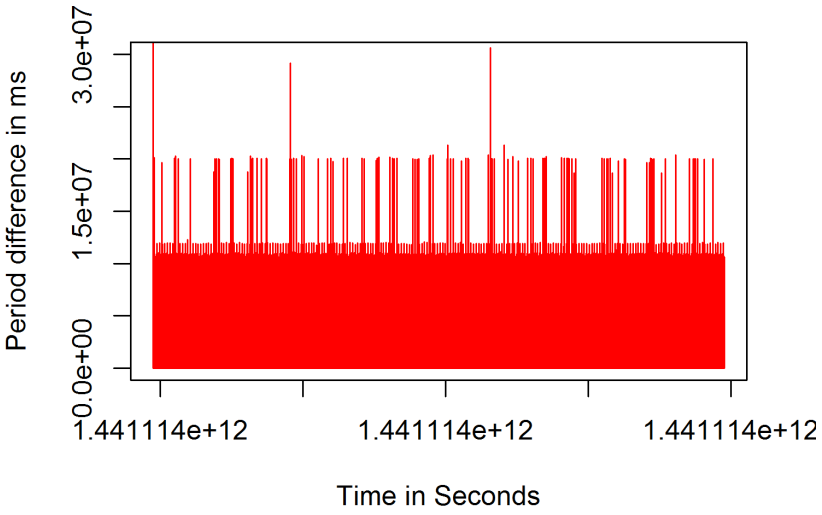

Figure 3.19: A plot produced by the data visualisation web app showing the variations in sampling period over time. It is clear that single samples are often missed and at times more than one.

3.7.1.6 Correlation

The cross-correlation of the raw and processed data is plotted, also indicating the delay between the datasets indicated by the largest cross-correlation value. This is used to ensure the raw and processed datasets are not time-shifted with respect to each other.

3.7.1.7 Time info

The difference between each consecutive *timestamp* and *system time* pair of any of the sensors can be plotted relative to the *system time*. This allows a visual comparison of the scale factor difference between the unknown timestamp unit and the known (ms) system time unit. Any inconsistencies in the sampling rate or missed samples are also clearly visible on these plots as shown by Figure 3.19.

3.7.1.8 Graphical interface for HMM parameter tuning

Due to the Bayesian nature of Hidden Markov models, they require a large amount of user interaction to find the global maximum solutions for the given system. This iterative process is greatly aided by the interactive tool developed for manually tuning selected parameters and visualising the resulting state sequence and the truth table comparing it to the ground truth data.

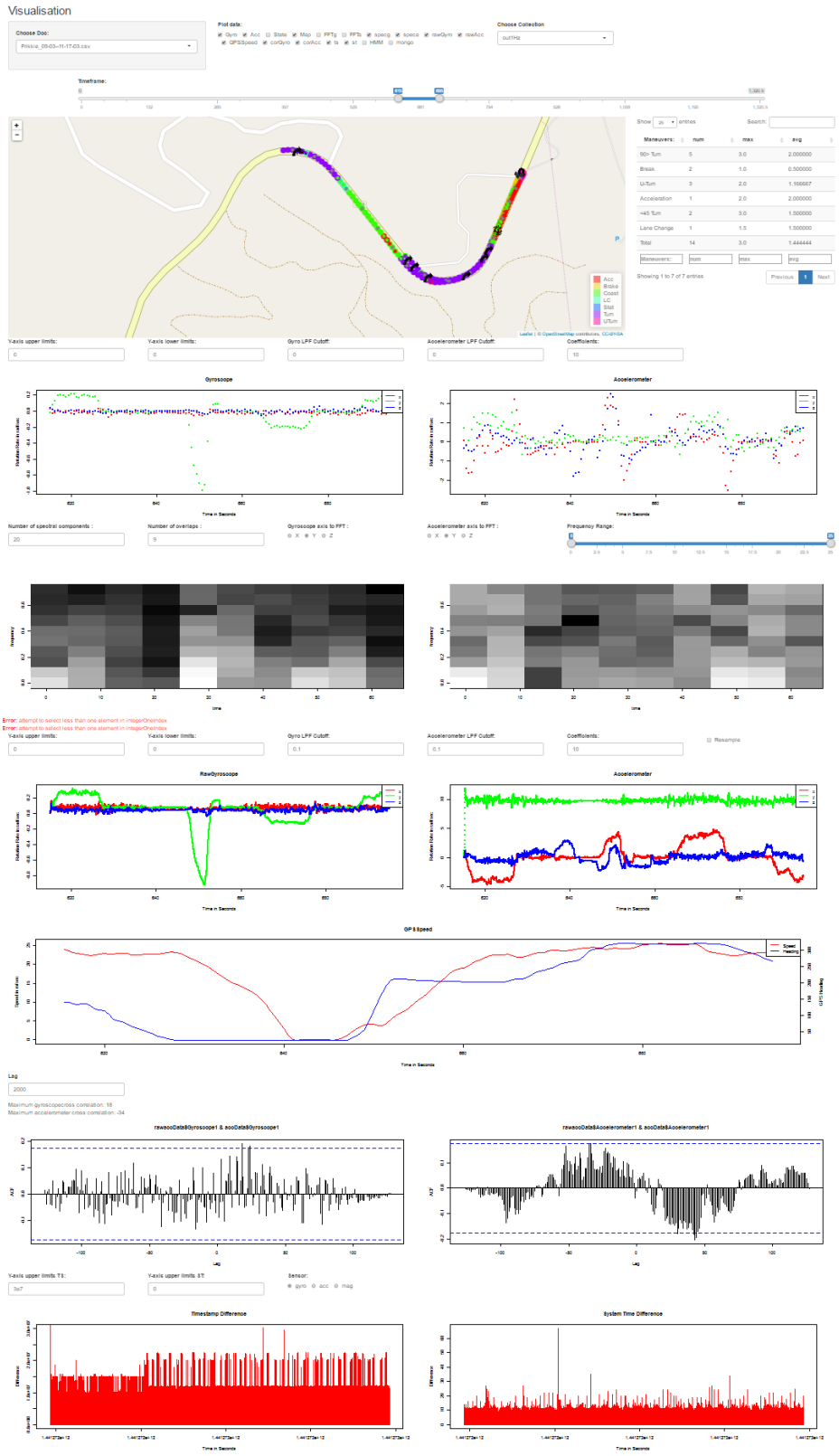

Figure 3.20: A screen shot showing an overview of some features of the developed R Shiny web-application. From top to bottom: 1. Data selection tools, 2. map and events summary table, 3. processed gyroscope and accelerometer plots, 4. spectrograms of processed gyroscope and accelerometer data, 5. unprocessed gyroscope and accelerometer plots with applied LPF, 6. GPS speed and heading, 7. raw and processed data cross correlation, 8. timestamp and system elapsed time between samples.

Chapter 4

Development tools and methodology

4.1 Introduction

Raw data acquired via a smartphone-based sensor, whether it be MEMS sensor- or GPS data, is prone to different types of errors. Additionally, the data acquired is not necessarily useful in its raw form. In the case of GPS; speed, heading and coordinate information is readily available at 1Hz, but acquiring information such as acceleration or rotation rate requires further processing. In the case of MEMS sensors, high sampling rate data is available, but extracting useful information from the overwhelming wealth of data provided by the sensors, necessitates further, application specific, processing.

The extraction of useful information is done by, among other techniques, performing: interpolation, multirate frequency conversion, quaternion reorientation, accelerometer gravity removal and spectral energy calculations. The literature on these techniques are discussed in Section 4.3. The mitigation techniques for errors encountered when sampling specific sensors from a smartphone are also discussed in this chapter. These errors include: accelerometer scale error, gyroscope bias,gyroscope scale error, various noise sources, sampling rate deviations and missing samples.

The chapter starts with literature on MEMS sensors and the challenges of sampling from MEMS sensors using a smartphone-based platform. A brief overview of the other literature relating to methods employed in the rest of the chapter is then given. In order to visualise the data initially, some processing has to be done. The minimal processing required to make the MEMS sensor data visually accurate and comprehensible is discussed in Section 4.4. An analysis then is performed to establish what information can be extracted from the higher frequency spectra in Section 4.5. The development of the complete algorithm to perform processing on the MEMS sensor data in accordance with objective 5 is described in Section 4.6, followed by a brief summary.

4.2 Micro electromechanical sensors

Microelectromechanical Systems (MEMS) are devices comprised of parts on a micrometer scale. The packaged devices are typically smaller than a few cubic millimetre in size [51]. Modern MEMS devices are packaged with semiconductor circuitry to perform accurate current or voltage measurements and quantisation of the signals produced by the micro electromechanical device and a form of inter-device communication (I^2C, SPI, etc.). In addition a MEMS product package often implements a form of filtering and error correction circuitry. The filter is designed to prevent aliasing and can often

be adjusted to the user's desired cut off frequency. The error correction circuitry is discussed in Section 4.2.2.

4.2.1 MEMS sensors in smartphones

The diminutive size and low energy consumption of these devices has allowed them to be widely integrated in mobile and battery-operated devices such as smartphones. Their production, however, is a specialised process and is currently outsourced to companies with expertise in the field. The Samsung Galaxy S7, for example, has an accelerometer and gyroscope from STMicroelectronics, a magnetometer from AKM Semiconductor and a barometric sensor from Bosch Sensortech. [52] . In most popular modern smartphones, the MEMS sensors are therefore one or multiple discrete components from different manufacturers than the phone manufacturer. This has multiple repercussions when using a smartphone as a sampling platform:

- Firstly, the timestamps that accompany a value read from a MEMS sensor's output register are not of a conventional format. Depending on the MEMS device being sampled, this output could be in milliseconds since the epoch, nanoseconds since application start or milliseconds since application start.

- Secondly, low-cost MEMS sensor ICs only provide one or two registers (8-16bit) for storing output data, these register values change at a rate matching the required sampling rate and if they are not read at an appropriate time by the Android application, duplicate - or missed samples result. This is where Android run time (ART) and its predecessor, Dalvik virtual machine (DVM), the process virtual machines that form part of the Android software stack, cause some problems. Due to the non-real-time nature of these operating systems, the reading of sensor data does not enjoy a high priority and is often delayed enough to miss reading a sample from a register before it changes.

Each data aggregation application used in this thesis records both the timestamp supplied by the relevant sensor as well as the "system time" when the interrupt to read the sensor value was triggered

Mitigating these flaws in the timing data is crucial, as no DSP can be attempted without uniformly sampled data or at least accurate times for each sample.

By recording both the system time and timestamps, corrections for some of these inconsistencies can be attempted.

4.2.2 MEMS sensor error models

The discrepancy between what a MEMS sensor measures can be modelled as a series of scale factor error multiplications followed by a series of additive error additions.

Typical MEMS devices may implement, among other corrections, a factory scale factor adjustment, a factory bias adjustment and, for temperature sensitive MEMS sensors such as a gyroscope, a thermometer-aided temperature induced bias compensation. The Android motion sensors API includes some additional error corrections.

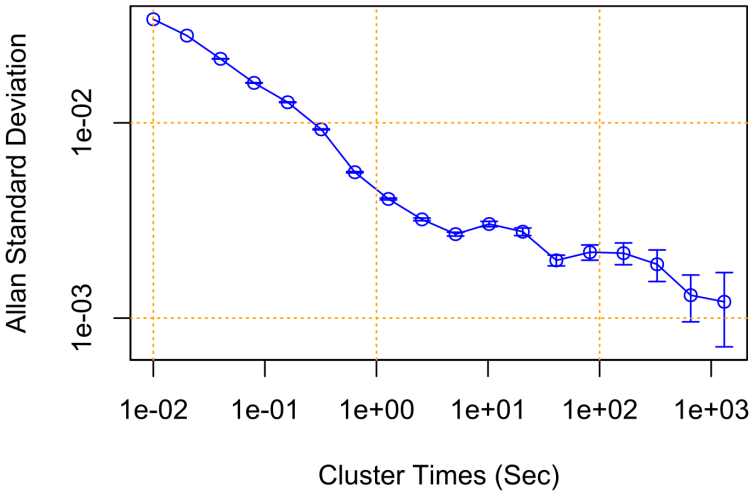

Figure 4.1: The Allan variance graph of a stationary smartphone-based MEMS gyroscope's x-axis.

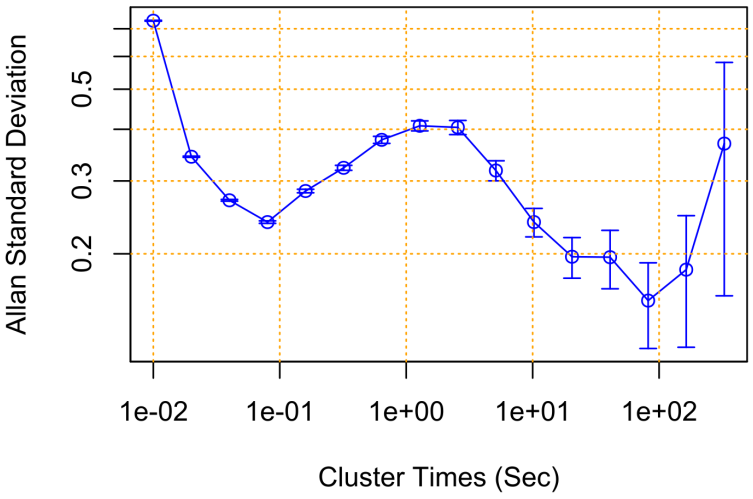

Figure 4.2: The Allan variance graph of a smartphone-based MEMS gyroscope's x-axis within a stationary vehicle with the engine running and people within the vehicle.

The gyroscope data obtained via the Android motion sensors API has additional gyroscope calibration applied in the form of online random walk bias estimation and correction. When data is collected from a perfectly stationary phone. This correction is clearly visible from the lack of long term bias error on the Allan variance graph as shown in Figure 4.1. The Allan variance shows the deviation of a measurement from a ground truth value at various frequencies. Figure 4.2, however, shows that the correction either does not take place or is ineffective when used in a vehicle affected by the movements of people within the vehicle or engine vibrations.

4.3 Literature on methods used

This section provides a brief overview of the knowledge needed concerning methods used in the rest of this chapter.

4.3.1 Interpolation

As shown Figure 3.19 the data collected in Section 3.4 has a considerable amount of missing samples. In order to obtain a dataset with a constant sampling frequency f_s three methods were compared i.t.o. their accuracy and efficiency..

4.3.1.1 Cubic spline interpolation

Polynomial splines of the 3rd degree are fitted through known points with the spline subject to constraints such as equality when passing over known points and gradient equality and second derivative equal to zero where different splines meet. This is a more computationally efficient interpolation method than sinc interpolation as the sinc function is also used as a sort of spline of infinite order compared to the lower order splines used for cubic spline interpolation. Causal implementation is more abstruse, but can be achieved with a few samples delay [53].

4.3.1.2 Linear interpolation

The most straightforward and computationally efficient interpolation method is linear interpolation, and can be causally interpolated with only 1 sample delay. The result is, however, not necessarily band-limited and requires a significantly higher than Nyquist sampling rate in order to produce accurate results. Interpolation errors tend to regress towards the signal mean.

4.3.2 Multirate filters

In order to enable data frequency conversion online and with limited processing power, efficient multirate signal processing tools are required. A multirate filter is a generic term for interpolation filters, decimation filters and combinations of interpolation and decimation filters.

Efficient multirate filters can be developed using the polyphase representation of the signals and filter impulse responses[54].

4.3.3 Eigen value decomposition

The Eigen Value Decomposition (EVD) performs an orthogonal transform on a set of multi-dimensional data. The goal is to remove correlation between axes by finding the axis with maximum possible variance followed by an axis orthogonal to the first with the maximum possible variance and the desired number of remaining orthogonal axes in decreasing order of variance. The original data can then be projected onto these orthonormal axes.

The orthonormal vectors can be obtained mathematically from the Eigenvalue Decomposition(EVD) or Singular Value Decomposition (SVD) of the covariance matrix of the data. The EVD is computationally more efficient for this application and is shown in Equation 4.1, where $\mathbf{X}$ is the zero mean data, $\mathbf{Q}$ is a square matrix where

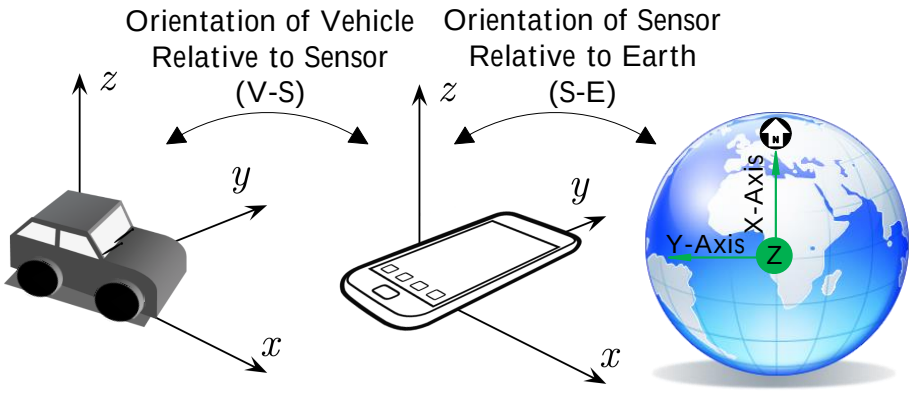

Figure 4.3: This figure shows the conventions used for the sensor- vehicle- and earth axes as well as the rotations between the axes.

Table 4.1: Summary of conventions used for x-,y- and z-directions in the earth-, vehicle- and sensor-axes

Axes	Direction		
	X	Y	Z
Earth	Magnetic North	Magnetic West	Upwards from surface
Vehicle	Front of vehicle	Left of vehicle	Top of vehicle
Smartphone	Right hand side facing screen	Top of device when facing screen	Pointing out of screen

each column is an eigenvector and the diagonal elements of $\mathbf{\Lambda}$ are the corresponding eigenvalues [55].

$$\mathbf{X}^T\mathbf{X} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1} \tag{4.1}$$

4.3.4 Rotational kinematics

The mathematics behind the orientation and changes in orientation of an object is called rotational kinematics. Three sets of axes are of importance in this thesis: the earth axes, the vehicle axes and the sensor axes. The standards defined in Table 4.1 are used to define the directions of the x-,y- and z-axis. Figure 4.3 shows the axes convention for the smartphone-, vehicle- and earth axes.

There are various methods of representing and performing operations with orientations, each with its distinct advantages and disadvantages.

4.3.4.1 Euler angles

Euler angles represent a rotation using three angles for three consecutive rotations in a conventional order. One convention would be to apply the angles yaw, pitch and roll in that order. This method is very intuitive, but has many drawbacks, including the jumps from 360 to 0 degrees during continuous motion, the "gimbal lock" problem (singularities at certain orientations where normal rules do not apply), the inability to arbitrarily combine two consecutive rotations and the computational inefficiency of operations. Figure 4.4 shows the convention used in this thesis for Euler angle direction and order.

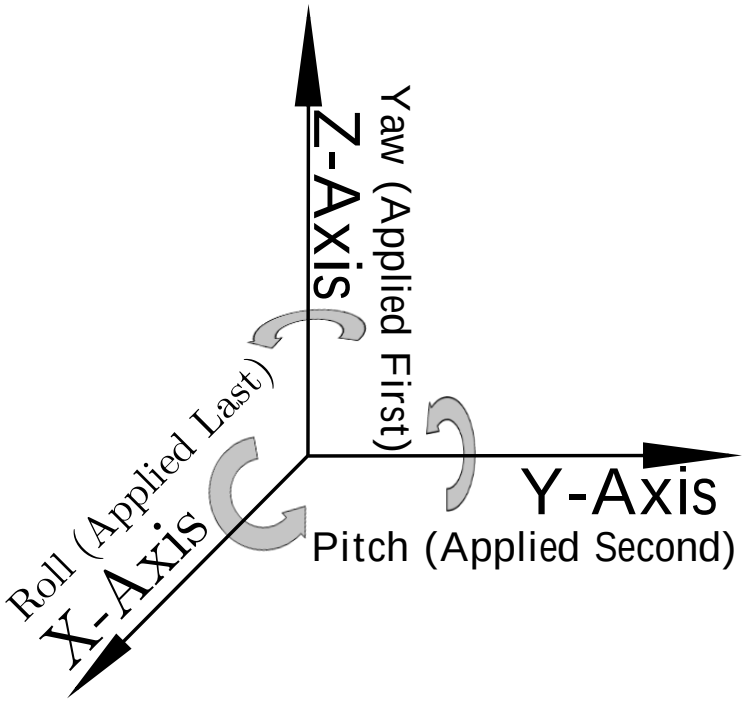

Figure 4.4: A diagram illustrating the conventional rotational direction and order for Euler angles.

4.3.4.2 Directional cosine matrix

The Directional Cosine Matrix (DCM) or Rotation Matrix is a 3×3 dimensional matrix that represents a rotation in 3D. The axis and angle method provides another way of portraying rotations. The DCM can be used to combine successive rotations and is not susceptible to the "gimbal lock" problems presented by Euler Angles or Axis and Angle methods. DCMs are, however, not trivial to understand and because they consist of nine elements, operations using DCMs are tedious and computationally inefficient.

4.3.4.3 Axis and angle

Any rotation in a 2D-plane can be represented by a single angle. A rotation in 3D can therefore be achieved by projecting onto a 2D-plane and then rotating by an angle on the 2D-plane. The orientation of the 2D plane can be represented by the unit vector orthogonal to the plane, this unit vector is the "axis" of the Axis and Angle rotation. This method is also rather trivial to understand, but still has most of the drawbacks listed for Euler Angles.

4.3.4.4 Quaternions

Quaternions are four-part mathematical constructs, subdivided into a scalar part ω and a three-dimensional vector part $\mathbf{v} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$. The quaternion can be written mathematically as follows (note the convention of positive complex numbers, this differs between sources) :

$$q = \omega + \mathbf{v} = [q_w, q_x, q_y, q_z]^T \tag{4.2}$$

$$q = \cos(\frac{\theta}{2}) + \mathbf{i}(x \times \sin(\frac{\theta}{2})) + \mathbf{j}(y \times \sin(\frac{\theta}{2})) + \mathbf{k}(z \times \sin(\frac{\theta}{2}) \tag{4.3}$$

Where i, j and k is described by:

$$i \times i = j \times j = k \times k = -1 \tag{4.4}$$

and

$$i \times j = -j \times i = k \tag{4.5}$$

Quaternions provide certain notable advantages over the other methods mentioned:

- Quaternions provide a way to represent rotations using 4 variables in an intuitive way(similar to Axis and Angle).
- Quaternions don not present any discontinuities or "Gimbal Lock" problems for 3D rotations.
- Quaternions can be used to represent and rotate a non-unitary 3D vector.
- Quaternions can be multiplied to combine two rotations into one quaternion.
- Quaternion operations are computationally efficient.

Useful quaternion operations include multiplication, conjugation, normalisation and rotation using the "sandwich product".

Multiplication Quaternion multiplication follows the same rules as complex multiplication using the identities in equations 4.4 and 4.5. If $q_1 = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$, $q_2 = e + f\mathbf{i} + g\mathbf{j} + h\mathbf{j}$ and $\otimes$ is a quaternion multiplication, then multiplication leads to the following equation:

$$z1 \otimes z2 = ae - bf - cg - dh + \mathbf{i}(be + af + ch - dg) + \mathbf{j}(ag - bh + ce + df) + \mathbf{k}(ah + bg - cf + de) \tag{4.6}$$

Conjugation The conjugate of a quaternion is simply the same quaternion with the complex parts negated.

$$q' = conj(a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}) = a - b\mathbf{i} - c\mathbf{j} - d\mathbf{k} \tag{4.7}$$

Normalisation The norm of a quaternion is the square route of the sum of the square of its components:

$$Norm = ||q|| = sqrt(q \times conj(q)) = sqrt(a^2 + b^2 + c^2 + d^2) \tag{4.8}$$

To normalise a quaternion, each element is divided by the norm so the new norm is equal to one.

Rotation The most important and useful quaternion operation is the rotation. Any coordinate or vector in 3D space can be rotated by writing the point and the rotation in quaternion form: $Point(x, y, z) = 0 + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$, $Rot' = conj(Rot)$ and $Rot = w + a\mathbf{i} + b\mathbf{j} + c\mathbf{k}$

$$RotatedPoint = Rot \otimes Point \otimes Rot' \tag{4.9}$$

This is called the "sandwich product".

Combining rotations Performing two consecutive rotations is a common occurrence when working with rotations in multiple frames of reference. Compound rotations can be created by combining two quaternions if q_1 represents the first rotation and q_2 the second, the compound orientation in the absolute frame of reference is given by:

$$q_{combined} = q_2 \otimes q_1 \tag{4.10}$$

The compound orientation in a rotated object's frame of reference would be given by:

$$q_{combined} = q_1 \otimes q_2 \tag{4.11}$$

If you have a quaternion representing the orientation of an arbitrary object 1 relative to earth (q_E^1) and the orientation of an arbitrary object 2 relative to object 1 (q_1^2) the orientation of object 2 relative to earth (q_E^2) can be calculated:

$$q_E^2 = q_1^2 \otimes q_E^1 \tag{4.12}$$

Note that the order of multiplication would be reversed if the negative complex number convention was used in Equation 4.3.

To euler angles Quaternions can be converted to Euler angles by using the following three formulas (assuming Euler angles applied in the order shown):

$$yaw = atan2(2 \times q_y \times q_w - 2 \times q_x \times q_z, 1 - 2 \times {q_y}^2 - 2 \times {q_z}^2) \tag{4.13}$$

$$pitch = asin(2 \times q_x \times q_y + 2 \times q_z \times q_w) \tag{4.14}$$

$$roll = atan2(2 \times q_x \times q_w - 2 \times q_y \times q_z, 1 - 2 \times {q_x}^2 - 2 \times {q_z}^2) \tag{4.15}$$

Exceptions to these equations occur when: $q_x \times qy + q_z \times q_w = 0.5$, in this case:
$yaw = 2 \times atan2(q_x, q_w)$
$roll = 0$
Another exception occurs when $q_x \times q_y + q_z \times q_w = -0.5$, then:
$yaw = -2 \times atan2(q_x, q_w)$
$roll = 0$

Quaternion between two vectors A quaternion (q_1^2) describing the rotation between two vectors, $\mathbf{v}_1$ and $\mathbf{v}_2$ can be calculated using a rotation around the vector given by the cross product of the two vectors:

$$q_{xyz} = v_1 \times v_2 \tag{4.16}$$

$$q_w = \sqrt{||v_1||^2.||v_2||^2} + (v_1 \cdot v_2) \tag{4.17}$$

Following these equations, the resulting quaternion $q_1^2 = [q_w, q_x, q_y, q_z]$ is normalised as shown in equation 4.8.

Detailed information on the derivations of equations in Section 4.3.4.4 as well as other useful quaternion equations can be found in online tutorials [56].

Spherical linear interpolation (SLERP) Performing only part of a quaternion rotation or finding the quaternion a fraction of the distance between two quaternions can be done using SLERP. Normalised linear interpolation is inaccurate for quaternions as the quaternion does not move with a constant velocity between two points. Equation 4.18 shows how SLERP can be performed, with: q_1 = the first quaternion
q_2 = the second quaternion
t = the scalar rotation fraction between 0 and 1
θ = half of the angle between the quaternions.

$$q_s = \frac{q_1 \sin((1-t)\theta) + q_2 \sin(t\theta)}{sin(\theta)} \tag{4.18}$$

4.3.5 Spectral DSP

Spectral analyses are useful for identifying frequency bands of interest when working with the 50-350Hz data provided by smartphone-based MEMS sensors. The tools used in this thesis to analyse and utilise the spectral information are described here.

4.3.5.1 Fast Fourier Transform

The discrete Fourier transform (DFT) is used to obtain the discrete complex frequency domain representation of a discrete time signal with equally spaced samples. A fast Fourier transform (FFT) is the efficient computer implementation of the DFT. The FFT has a computational complexity of $O(n \log n)$ compared to the $O(n^2)$ of the DFT. The computational efficiency of the FFT allows it to be implemented online on an Android phone using one of several available libraries.

$$\mathbf{X}_k = \sum_{n=0}^{N-1} x_n . e^{-2\pi i k n / N} k \in \mathbf{Z} x \in \mathbf{R} \tag{4.19}$$

The FFT can be applied to multi-dimensional data such as the data from a three orthogonal-axis MEMS sensor as three discrete FFTs.

4.3.5.2 Spectrogram

A spectrogram is a representation of the frequency spectrum of a signal as it varies with time. Typically visualised with 3-axis graphs with one time-axis, one frequency axis and one amplitude axis. The amplitude axis is often represented by a colour scale.

Intermittent repetitive noise can be identified by seeking areas on the spectrogram with high spectral densities. The noise can also be characterised by its frequency domain location, variations in frequency over time and variation in amplitude over time.

A spectrogram is produced by performing a FFT on a sequence of time domain signal windows which can vary in sample length, shape and overlap. Resolution and spectral leakage are influenced by the size and type of window used to obtain time series data of a finite extent. A longer window increases frequency resolution, while a higher window to signal duration ratio (more overlap or shorter window) increases time resolution. The shape of the window mainly dictates the amount of undesirable spectral leakage experienced.

4.4 Minimal processing

In order to initially view the recorded data and perform elementary visual inspections of data quality or qualitative comparisons, a number of essential processing steps need to be taken. This processing is done in order to extract visually accurate information from raw data with minimal processing while preserving the largest possible amount of data.

4.4.1 Data separation

The logging application often reads multiple data-points during a single interrupt. These samples are then associated with the same timestamps and system time. At this point each single CSV file, containing the values for all sensors, is split into 8 CSV files representing each of the sampled sensors.

4.4.2 Time correlation

The temporal nature of the data means that the timestamps paired with the data points are of particular value. Due to the problems stated in Section 4.2.1, however, the use of sensors' timestamps require a large amount of complex processing. The Android UNIX system time is recorded by the logging app at the moment that the sampling interrupt is triggered. In order to obtain a passible degree of accuracy with minimal processing, this time value is used.

In order to incorporate the flags from the test done in Section 3.4.3, the sorted data from each sensor and each recording session is binary searched for the closest matching system time of each flagged event. The flag is then associated with the relevant sensor-sample. There were no duplicate flags, but in the case that multiple flags were close to a single data-point, the closest point was attached to the sample in question and subsequent flags to the consecutive points.
The result of this processing is a series of database collections, containing flagged data points of a specific sensor's data as well as flagged GPS data. For example: a data-sample could consist of gyroscope - and an accelerometer data associated with the same system time.

4.4.3 Noise filtering

In order to improve the clarity of the data when plotted, the data is low pass filtered to 2Hz with a Butterworth filter. (The choice of frequency was experimented with and iteratively optimised to reduce noise here, but choice of frequencies is discussed in Section 4.5.1) This removes high frequency noise caused by road vibrations and vehicle engine noise. In order to do this, the approximating assumption is made that the data is uniformly sampled. This assumption is inaccurate for quantitative data as the time difference between data-points varies due to missing samples, etc. (Discussed in Section 4.2.1). However, for the qualitative and visual comparison, this method is adequate. The resulting data is referred to in the data visualisation web app and code as the "Raw Data".

4.5 Non-causal analyses

The following processing was not done with causality or computational efficiency in mind, but was used merely as analytical tools to extract information from the aggregated data.

4.5.1 Spectral information analysis

The MEMS sensors on Android based smartphones can sample at rates in excess of 100Hz. According to the Nyquist criterion periodic signals of up to 50Hz can therefore be reproduced without aliasing. In practise, due to missing samples, this rate is significantly lower. Sampling at the highest possible frequency and processing data at such a high rate has undesirable effects in terms of computational expense and energy efficiency. Therefore it was important to identify at which frequencies relevant information, w.r.t. the driving events which are to be detected, can be located.

4.5.1.1 Events with low observability and stationarity.

Three of the events chosen in Section 3.2 have low observability and stationarity when detected via GPS, necessitating the use of higher frequency information from the MEMS sensors. Speed bumps, swerves and lane change tests were conducted to determine the data rates required for detection. The required data rate was found to be around 2Hz, which was the minimum rate to observe swerves and lane-changes.

4.5.1.2 Vibrations

The accelerometer and gyroscope datasets were analysed via spectrogram in order to distinguish between different vibrations measured by these devices. The spectrogram developed for the data visualisation web-app described in Section 3.7.1.7 used with a default Hamming window of 40 samples with 20 samples overlap provided adequate frequency- and time-domain resolution.
Two primary sources of vibration were identified: Vibration caused by the vehicle's engine and vibration caused by the vehicle's interaction with the road surface, causing high frequency vibrations that are not filtered out by the suspension. In order to pinpoint the frequencies where these vibrations occurred, the GPS speed was used as an indicator of stationarity to correlate the amount of spectral energy in a chosen frequency segment with the vehicle's movement. The 4-7Hz range was found to have the highest correlation between spectral energy and vehicle movement. The 18-22Hz range was found to contain the largest amount of vibration caused by an idling engine.

4.6 Causal processing

A recurring theme in Chapter 1 is the "smartphone-based" nature of the algorithms to be developed. This entails algorithms that are not too computationally demanding to be run live on a smartphone device. All algorithms in this section were designed to be causal and as computationally efficient as possible. Figure 4.5 shows a simplified overview of the processing pipeline developed in this section.

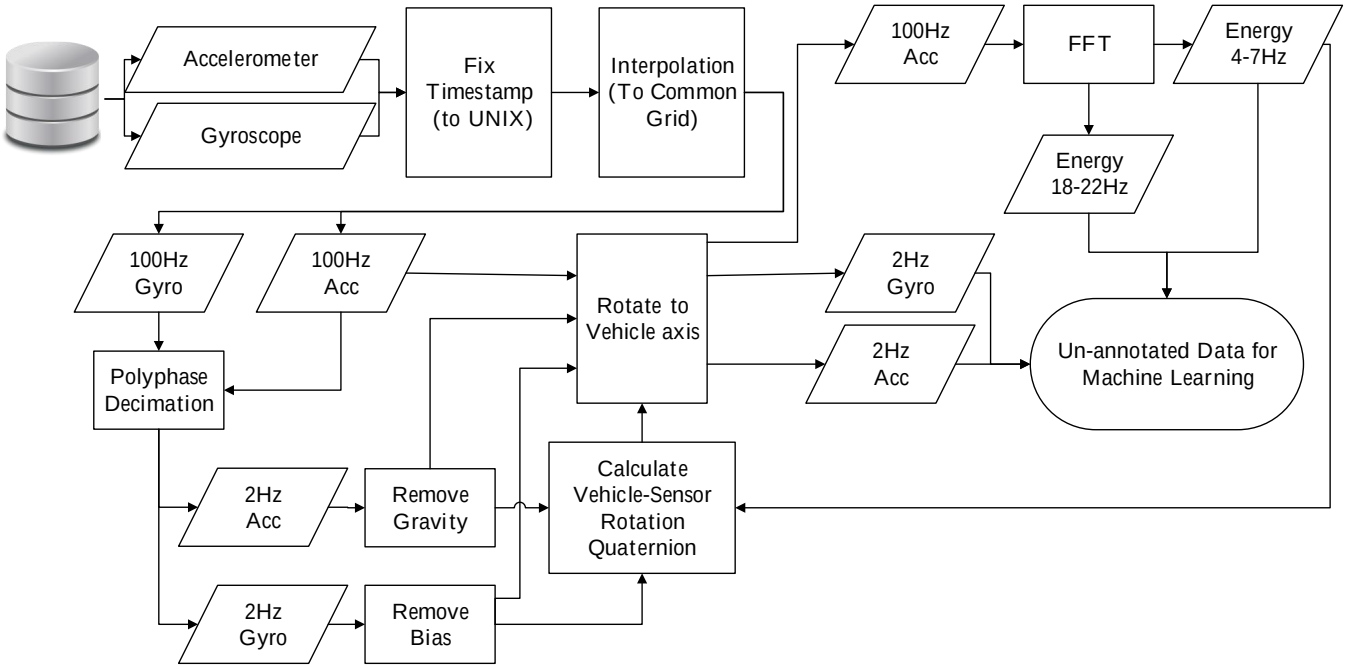

Figure 4.5: An information-flow diagram of the developed data processing pipeline.

4.6.1 Interpolation

In order to produce a uniformly sampled dataset, three interpolation methods were considered: Sinc interpolation, cubic spline interpolation and linear interpolation.

Sinc interpolation produced adequately accurate results when used offline, but could not be implemented causally. The spline interpolation results were extremely sensitive to deviations in timestamp accuracy and caused large erroneous surges in the data. Linear interpolation was eventually chosen, because the sampling rate of the data was significantly higher (double) than the required Nyquist rate, errors regressed towards the mean and the deviations in timestamp accuracy did not lead to erroneous spikes in the data.

4.6.2 Sampling rate conversion

One of the disadvantages of using a smartphone as a sensing platform is that all processing comes at a cost. Not only does increased processing affect the battery life, if more processing power is required than a phone can provide, information might be lost (skipped samples). MEMS sensor data can be sampled at rates of up to 200Hz on Android phones, but this does not necessarily improve the overall accuracy. In order to avoid unnecessary processing of data, the frequencies containing relevant information must be identified and a data rate must be chosen to correspond to the Nyquist frequency (twice the maximum required frequency information). Analyses were done in Section 4.5.1 correlating the type of - and severity of an event with the energy in its various frequency bands. While the majority of information was found at low frequencies, the higher frequency ranges were also found to contain some useful information. Two data rates were therefore chosen, a lower (2Hz) and higher (100Hz) frequency.

The higher frequency comes as the result of the interpolation in Section 4.6.1, while the lower frequency is obtained by decimating the uniform 100Hz signal and passing it through an efficient polyphase LPF as discussed in 4.3.2.

4.6.3 MEMS sensor calibration

The calibration of the MEMS sensors presented a significant obstacle for this thesis. The overarching aim is to allow the app te be seamlessly downloadable and usable on any Android device with the prerequisite sensors, therefore complicated calibration, processes that require driver input or processes that require additional hardware are to be avoided. The noise model of the MEMS inertial sensors is discussed in Section 4.2.2. Taking into account the basic noise model and the compensation that the Android sampling APIs perform, the following variables need to be calibrated for each individual smartphone running the application. Some of these variables can change over time and are indicated as such:

- Gyroscope bias (Slowly time varying)
- Gyroscope noise standard deviation.
- Accelerometer scale error
- Accelerometer noise standard deviation.

Because vehicle stationarity can be detected by GPS, or by the lack of certain vibration frequencies in the accelerometer, various calibration processes can be performed given the knowledge that the vehicle is stationary.

Without the knowledge that the sensor is being used within a vehicle, however, the smartphone is unable to accurately calculate the bias due to the minor accelerations measured even when the vehicle is stationary as deduced from the Allan variance graphs in Section 4.2.2.

Correctly scaled data with the bias removed is only of interest for the low frequency (2Hz) data-stream as the high frequency (100Hz) stream is only analysed in terms of its frequency content and the relative amount of variation along the 3 axes. Therefore, the calculation of all listed calibration variables is done using the LF MEMS sensor data and the corrections are only applied to the 2Hz data stream.

Figure 4.6 describes the calibration of the gyroscope and accelerometer sensors:

1. Calibration occurs when road-vibrations acceleration and – if applicable– GPS speed to fall below the thresholds for stationarity set in Section 4.7.2.
2. Gyroscope and accelerometer readings are recorded in 5 second overlapping windows. (10 seconds is shown by the Allan variance analysis to be the time period with the lowest short-term variation from the truth, as well as being a long enough period to ensure stationarity.)
3. The window with the lowest standard deviation for the weighted sum of sensors over a 10 second period is selected.
4. The noise standard deviations for both sensors are updated if the measured standard deviation is lower than previous measurements.

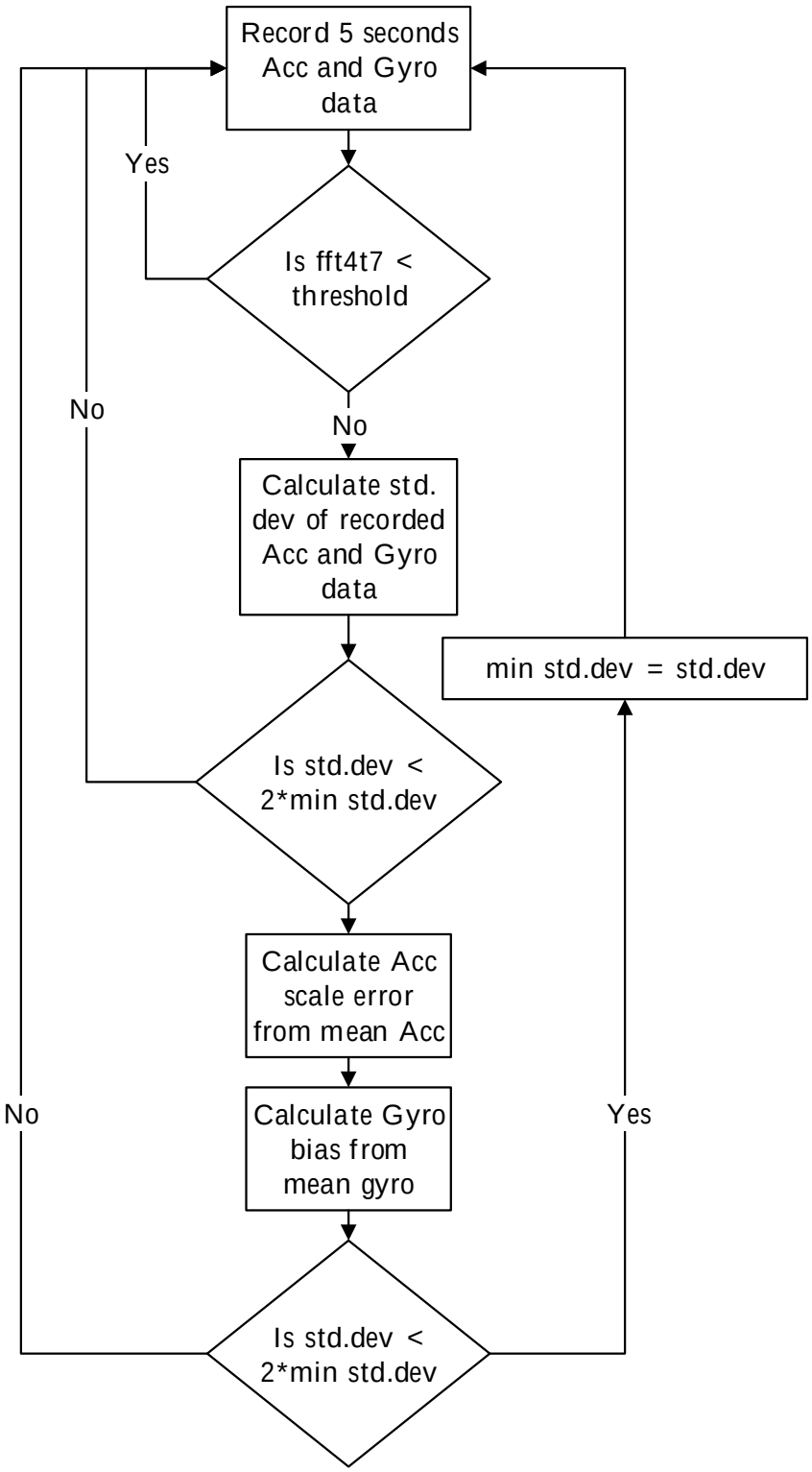

Figure 4.6: A flow diagram describing the MEMS sensor calibration process.

5. The mean of periods where the standard deviation is below twice the lowest standard deviation and the vehicle is stationary are used as the bias for the gyroscope .

6. The Euclidean norm of the accelerometer reading, when the vehicle is determined to be stationary and the standard deviation is below twice the lowest standard deviation, is used to estimate the scale factor error w.r.t. gravitational acceleration of $9.8m.s^{-2}$.

The following calibration variables are not calculated, since the benefit they would provide are estimated to be negligible for the current application: Varying scale factor differences between MEMS sensor axes, gravitational effects on the gyroscope, gyroscope and accelerometer misalignment, gyroscope scale error and accelerometer bias.

4.6.4 Reorientation algorithm

In order to address the problem presented by sampling from a mobile MEMS sensor device within a moving vehicle (Objective 5), a novel reorientation algorithm is designed. The essence of the problem is knowing the orientation of the vehicle relative to the sensor. This is not to be confused with the gravity removal algorithm (Section 4.7.1), which requires a knowledge of the orientation of the sensor with respect to the gravity vector (up) as shown in Figure 4.3.

Sections 4.6.4.1 through 4.6.4.3 describe the sources of known vectors that can be used to estimate the orientation of the smartphone relative to the vehicle, while section 4.6.4.3 is concerned with deriving a quaternion orientation given two vectors.

4.6.4.1 Axes of most and least variation.

The spectral information analysis of Section 4.5.1 revealed unique frequency characteristics that can be exploited in order to determine the orientation of the sensor relative to the vehicle. The spectral information analysis in Section 4.5.1 revealed that there were significantly less low frequency ($<$1Hz) variation in z-axis acceleration of a vehicle when compared to the other axes. This is easily explained as a vehicle has very limited freedom of motion (bounded by the suspension characteristics) in the axis normal to the road. It was also clear that the y-axis (lateral axis) of the gyroscope showed significantly more high frequency variation than the other axes.

In order to find the accelerometer axis with the least variation, the covariance matrix (or rather a matrix which is proportional to the covariance matrix) of a windowed portion of zero mean low pass filtered accelerometer data is calculated using Equation 4.20, where $\mathbf{X}$ is a n by p matrix where n is the number of samples and p the number of dimensions. The Eigen Value Decomposition (EVD) of the covariance matrix is calculated yielding the squared eigenvalues (σ) of the data and the associated eigenvectors (Q). The eigenvector corresponding to the lowest eigenvalue is the represents the required axis of least variation in the sensor axes $V_s^{vertical}$.

$$(X - \bar{X})^T (X - \bar{X}) = Q\sigma Q^{-1} \tag{4.20}$$

This process gives the axis of least variation ($V_s^{vertical}$), however, the correct hemisphere of the vector and consequently the z' vector ($V_s^{z'}$) has not been determined. This is done by calculating the dot product of the obtained vertical axis ($V_s^{vertical}$) and

the mean acceleration over the windowed period as shown in Equation 4.21. If the dot product is positive, the vector is in the correct hemisphere and $V_s^{vertical} = V_s^{z'}$. If the dot product is negative, the vector needs to be rotated by 180 degrees as shown in Equation 4.22.

$$cos\theta = V_s^{vertical} \cdot \bar{X} \tag{4.21}$$

$$V_s^{up} = V^v ert_s \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{4.22}$$

The process is repeated in order to find the lateral axis, but with a few changes. The unfiltered (high frequency) gyroscope data is used and the axis of maximum variation (the eigenvector corresponding to the largest eigenvalue) is used. The hemisphere correction is done using the forward vector given in Section 4.6.4.2. Testing showed that the results of this reorientation technique, especially when using high frequency information is to a large extent dependent on the dampening characteristics of the device mount.

4.6.4.2 Forward vector in a turn

Another method used for determining the forward axis of the vehicle is the cross product of the gyroscope and accelerometer measurements. In the case of an anti-clockwise turn, the rotation vector is in the direction of the positive z-axis and the vehicle's lateral acceleration is in the direction of the negative y-axis. The resulting cross-product is therefore in the direction of the vehicle's forward axis. The same holds true for a clockwise turn, the rotation vector is in the direction of the negative z-axis and the vehicle's lateral acceleration is in the direction of the positive y-axis (left). The resulting cross-product is therefore in the direction of the vehicle's forward axis.

4.6.4.3 Quaternion calculation

At this point a few methods are described in literature to calculate a quaternion rotation between two sets of axes. One featured method uses an iterative method to estimate the quaternion that produces the smallest error in rotation [57] in this case, the quaternion property of multiplicative combination of rotations (Equation 4.11) is exploited.

A quaternion describing the rotation from the sensor axes to the vehicle axes, q_s^v, can be obtained from the estimated x-, y- and z-axis vectors in the sensor axes ($V_s^{x'}$, $V_s^{y'}$ and $V_s^{z'}$,), by calculating the two quaternions ($q1_s^v$,$q2_s^v$) to align two of these vectors with their respective vector in the vehicle axes ($V_v^{x'} = [1,0,0]$, $V_v^{y'} = [0,1,0]$ and $V_v^{z'} = [0,0,1]$).

The quaternion representing the shortest rotation between two vectors is given by equations 4.16 and 4.17. The vector used to calculate the second quaternion rotation ($q2_s^v$) needs to be rotated by ($q1_s^v$) before the calculation is done.

Finding the top of the vehicle is generally easier and more accurate than finding the front, therefore, the vector in the horizontal plane used to calculate $q2_s^v$ is first projected to the plane orthogonal to the estimated vertical vector using equation 4.23.

$$V_{proj} = V_{horiz} - \frac{V_{horiz} \cdot V_{vert}}{V_{horiz} \cdot V_{horiz}} . V_{vert} \tag{4.23}$$

These two quaternions can then be combined using Equation 4.11 to form q_s^v. This is equivalent to performing two consecutive rotations and aligning two respective vectors. The quaternion q_s^v can now rotate any MEMS sensor data from the sensor-axes to the vehicle axes using a simple multiplication defined in Equation 4.9.

4.7 Algorithm design

The current quaternion orientation estimation is updated when either the axes of most and least variation vectors (shown in Section 4.6.4.1) are available, the forward vector calculated by cross product is available (shown in Section 4.6.4.2), the vehicle is accelerating after being detected to be stationary (shown in Section 4.7.2) or when the phone is detected to be rotating at a rate that is faster than typical vehicle rotation (1 rad/s around the z-axis or 1.2 rad/s around the x- or y-axes). All orientation updates, except when high rotation rate is detected, are applied via a negative feedback loop implemented using quaternion SLERP, as shown in Figure 4.7. When high rotation rate is detected however, the gyroscope reading is integrated over time resulting in a quaternion representing the change in orientation of the phone. This rapid change quaternion (q_r) is quaternion multiplied with the current orientation quaternion ($q_s^v(t-1)$, resulting in a new orientation quaternion($q_s^v(t)$).

4.7.1 Gravity removal

The accelerometer features prominently in most proposed vehicle monitoring and reckless driving detection systems [58, 59, 60, 61]. In vehicle monitoring, it is imperative to distinguish between accelerating, decelerating and lateral coordinate acceleration, since this data can be used to identify driving manoeuvres and events effectively. Accurate coordinate acceleration data is therefore the main concern of this section.

> we [...] assume the complete physical equivalence of a gravitational field and a corresponding acceleration of the reference system. – Einstein.A 1907

Albert Einstein's equivalence principle states that one cannot distinguish between acceleration caused by gravity and acceleration caused by external forces. Accelerometers measure what is known as *proper acceleration* (the acceleration relative to a free falling point), not *coordinate acceleration* (the acceleration relative to a stationary point). The proper acceleration of an object which is stationary with respect to the earth will therefore be 1 g ($1 \times 9.8 m.s^{-2}$) upwards in the earth axes.

Work was done by the author (Bruwer et al.[62]) to develop a system that utilises a Kalman filter and the limited dynamics of a vehicle to remove gravitational acceleration from the accelerometer measurement. This system was, however, designed for a sensor placed in a fixed, arbitrary position within a vehicle before initial vehicle acceleration. For a device that can be moved to an arbitrary position within the vehicle at any time during driving, this algorithm becomes tough to implement and computationally expensive.

The Android Motion Sensors API provides implements a "Gravity" sensor which estimates the direction and magnitude of gravity (with other external accelerations removed) and a complementary "Linear Acceleration" sensor which estimates the acceleration of the sensor with the effects of gravity removed. These "software sensors"

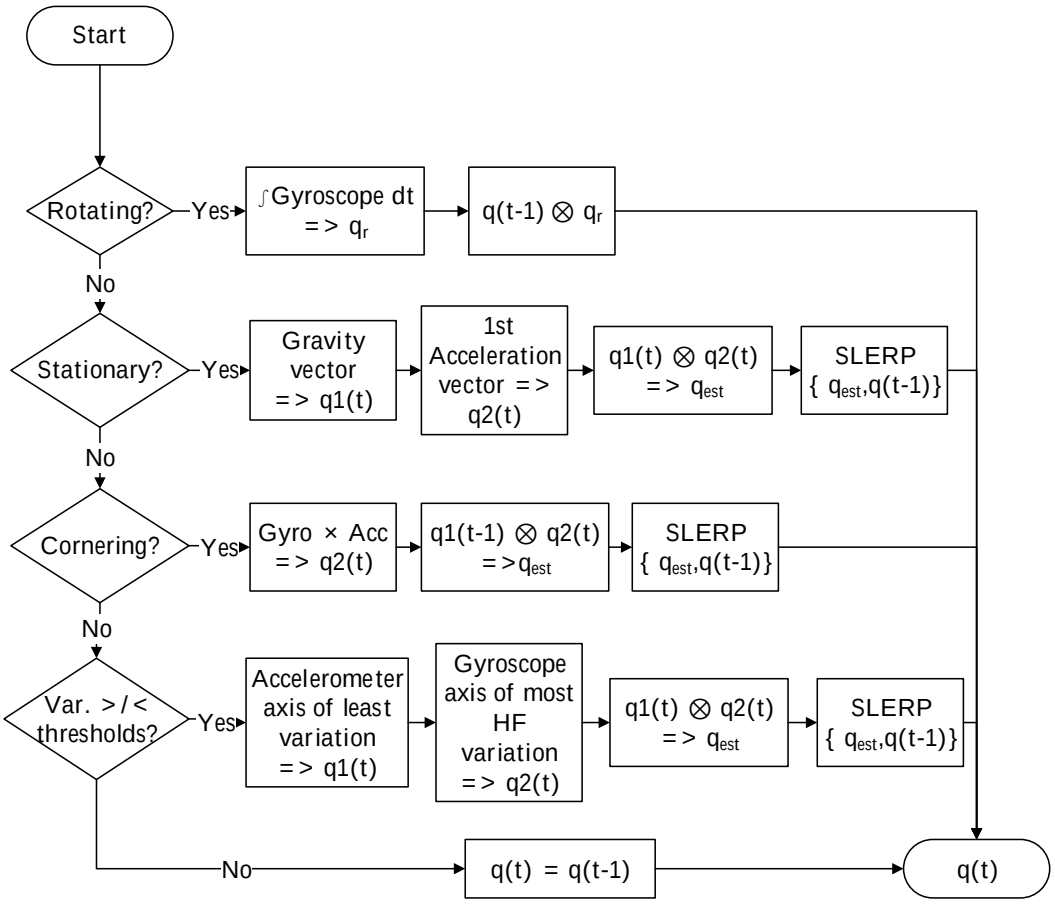

Figure 4.7: A flow diagram presenting a simplified overview of the reorientation algorithm's operation.

make use of a combination of high pass- and complementary filters to combine accelerometer and gyroscope information [63]. These algorithms are susceptible to inaccuracies during sustained acceleration and during testing, were shown to be ignorant of scale errors in accelerometers, always subtracting the theoretical magnitude of gravity.

In order to utilise the computationally efficient "software sensors", but avoid the errors in measurement caused by accelerometer scale error, an algorithm was implemented that uses the result of the "Gravity" sensor to continuously calculate a quaternion representing the orientation of the sensor relative to the earth's gravity vector q_e^s. The average acceleration recorded when the vehicle is detected to be stationary (Section 4.7.2) and is used as the magnitude of the gravity vector in the earth axes V_e^{grav}. The gravity vector is then rotated from the earth axes to the sensor axes using equation 4.9.

4.7.2 High frequency activity detection

Three valuable pieces of information can be extracted from high frequency MEMS sensor data as discussed in Section 4.5.1. Vehicle stationarity, engine vibration and the axis of most high frequency rotation (discussed in Section 4.6.4). Stationarity (road

noise) and engine vibration are both obtained by calculating the amount of energy in the specific frequency ranges determined to contain the respective vibrations. When the phone is in a vehicle with the engine running, the 18-22Hz range has a measurably higher amount of spectral power than when no engine noise is present. When the vehicle is moving, a measurable amount of spectral power is present in the 4-7Hz range.

To perform the road noise and engine noise calculations online, the HF accelerometer stream rotated to the vehicle axis (method discussed in Section 4.6.4) is used. The incoming data is 60 sample hamming windowed with 35 sample overlap and the FFT is calculated for each window. The band limited spectral energy for each window is calculated and this spectral energy value is compared to a threshold in order to obtain two boolean values indicating whether road noise or engine noise are present.

4.8 Summary

This section was primarily concerned with the processing of MEMS sensor data in accordance with objective 5 and thus starts by discussing MEMS sensors in a general sense as well as with specific reference to smartphone-based MEMS sensors. Literature necessary for was then provided, including a thorough overview of the possible methods that could be used for working with- and representing the rotational kinematics of the smartphones rotation within a moving vehicle. A short section is dedicated to the minimal processing necessary to make the raw data useful. A spectral information analysis was then performed and frequencies were identified for detecting vehicle movement and engine vibrations. The frequency domain analysis also helped isolate the minimum sampling frequencies at which faster manoeuvres can be detected. The complete algorithm was then assembled from its primary parts: the interpolation, sampling rate conversion, MEMS sensor calibration and the reorientation algorithm. The result of the processing described in this chapter is an un-annotated 2Hz dataset that can now be annotated and used for machine learning in the next chapter.

Chapter 5

Event Identification Methods

5.1 Introduction

The work in previous chapters details the development of a method for obtaining thoroughly descriptive data from smartphone-based MEMS sensors. This data, though containing vast amounts of detailed information pertaining to vehicle movement, is not intuitively comprehensible as it contains too many dimensions per unit time step and is fully numeric. In this chapter, steps are taken toward the fulfilment of objectives 5 and 6 from Section 1.4 by deriving a linguistic event-type for each sample of the time-series data. As discussed in Section 3.2, the events to be identified are:

- Acceleration
- Braking
- Turns
- U-Turns
- Lane Changing
- Swerving
- Speed Bumps
- Rough Road Surfaces
- Coasting
- Stationarity

Below follows an overview of the literature on the methods used for performing time series classification, two of these methods, Hidden Markov Models and Random Forests, are implemented and evaluated using training and testing labels. The process developed for annotating the acquired dataset with labels is also discussed in this chapter.

5.2 Literature on methods used

Time series classification is a large field of study, with various applications and corresponding techniques. Stock market predictions, population growth modeling, and speech processing are some of the more popular applications. Time series classification with multivariate continuous data and non-binary labels however, is a less pronounced area of study and proved to be a challenging subject.

Classifying time series data-points individually opens doors for more generic, off the shelf algorithms, but consequently information w.r.t. the time dependent nature of events is lost. The Hidden Markov Model provides a middle-ground by observing a single sample at a time while still taking into account the time dependence of transitions between classifications. For this reason, Hidden Markov models are also discussed in detail in this chapter. The three "off the shelf" machine learning methods implemented were: Random forests, Neural Networks and Boosting. Of these, Random Forest proved to be the best performer and was therefore analysed more thoroughly and discussed in Section 5.2.4.

5.2.1 Varying length data classification

When faced with the problem of multivariate continuous time series data, sources in the literature often resort to performing segmentation of the data prior to attempting machine learning, or to viewing the data as discrete independent samples and classify them as such [64]. Methods used for segmentation vary, but simple thresholding techniques with the thresholds empirically determined are common. Learning these thresholds from annotated data is possible, however, the complexity of the boundaries between data segments are still very limited. In the case of the MEMS sensor data used in this thesis, segments will also, necessarily, be of arbitrary lengths as the duration of events can vary from brief swerves to long turns or periods of coasting. Parametrisation and Dynamic Time Warping (DTW) are usefull techniques for performing machine learning on arbitrary-length data.

5.2.1.1 Dynamic Time Warping

When a valid driving event has been detected, the signals recorded during the event are compared to a set of templates using the dynamic time warping (DTW) algorithm [65]. DTW finds an optimal alignment between two time-dependent sequences with different lengths. Consider a matrix of the Euclidean distance between each point of two sequences, as seen in Table 5.1. Both sequences start at the bottom left corner. An optimal warping path constitutes the minimum sum of distances, while adhering to monotonicity, boundary and step size conditions. The template with the lowest minimum-distance warp path to the detected event is the closest match.

DTW has the advantage, in terms of this project's data, that it is the most accurate and flexible measure of similarity between two time-series of varying length. The disadvantages of DTW, however, are:

- DTW is a computationally expensive operation.
- DTW requires accurately segmented data with each segment representing a separate event.

Table 5.1: DTW cost matrix showing optimal warping path. Table adapted from article by Engelbrecht et al.[11].

Template	Minimum-distance									***10***
0	0	-1	-2	-3	-4	-4	-2	-1	-1	***0***
1	1	0	-1	-2	-3	-3	-1	***0***	***0***	1
2	2	1	0	-1	-2	-2	***0***	1	1	2
4	4	3	2	1	***0***	***0***	2	3	3	4
3	3	2	1	***0***	-1	-1	1	2	2	3
3	3	2	***1***	0	-1	-1	1	2	2	3
1	1	***0***	-1	-2	-3	-3	-1	0	0	1
0	***0***	-1	-2	-3	-4	-4	-2	-1	-1	0
Measured	**0**	**1**	**2**	**3**	**4**	**4**	**2**	**1**	**1**	**0**

- Choosing templates that are representative of the events to be detected is challenging.

5.2.1.2 Feature Extraction

Apart from DTW, another method for comparing time series data segments of unequal lengths is the parametrisation of or feature extraction from each segment. Given a varying length time series data segment, a set of parameters describing the set can be extracted from each dimension of the segment. Commonly extracted parameters include:

- FFT coefficients
- Wavelet Transform Coefficients
- Curve fitting parameters
- Minima, Maxima, Mean and other statistical parameters
- Signal Energy
- etc.

Choosing the most relevant features to extract is the challenging aspect of feature extraction, as too little information extracted will result in lost information and too much information extracted will result in a computationally expensive algorithm and increase the chances of over-fitting. Machine learning algorithms such as Random Forest gives feedback regarding variable importance for making a classification, enabling iterative adjustment of the parameters included.

5.2.2 Fuzzy logic

Fuzzy logic is a useful tool for making decisions based on rules that are explicit, yet in some sense ambiguous. These rules are specified by the user and Fuzzy Logic is

therefore a form of "Expert System". Rules are specified in a semi-human language, which is converted to its mathematical equivalent. Where the truth of rules in a standard conditional rule-based system is binary, the answer to a fuzzy system of rules is a degree of truth [66].

5.2.2.1 Universe of discourse

The universe of discourse is the numerical domain within which numeric inputs to the fuzzy system are allowed to vary. The range must be chosen to correspond with the range of numeric inputs given to the system and the granularity to the desired accuracy.

5.2.2.2 Operators

The only fuzzy operators used in this thesis are the "AND", "OR" and "NOT". Though all these are well known in boolean logic, the mathematical evaluation differs for fuzzy systems as well as between different fuzzy systems. The most commonly used evaluations, and the ones used in this thesis, are called Zadeh operators [67] as defined in Equation 5.1.

$$\begin{aligned} \text{x AND y} &= \text{minimum(truth(x),truth(y))} \\ \text{x OR y} &= \text{maximum(truth(x),truth(y))} \\ \text{NOT x} &= \text{(1 - truth(x))} \end{aligned} \tag{5.1}$$

5.2.2.3 Variables

A degree of membership function needs to be set up for each input (e.g. acceleration, speed, rotation) and output (e.g. turning, braking, swerving) variable with different linguistic levels (e.g. none, low, high) represented by a combination of functions (e.g. Gaussian, trapezoid, cone) as shown in Figure 5.3.

5.2.2.4 Rules

A linguistic rule is defined for each level of an output variable. The rules consist of input variables and operators in the form shown in Equation 5.2

$$\begin{aligned} \text{IF variable\_1 IS level\_1 OPERATOR variable\_2 IS level\_2 THEN} \\ \text{out\_variable IS level} \end{aligned} \tag{5.2}$$

5.2.2.5 Evaluation

Fuzzy inference is performed to create a generalised fuzzy set using the inputs, defined membership functions and rules. The generalised fuzzy set can be visualised as the degree of membership of the inputs to each value in the universe of each variable. This generalised set can then be defuzzified, using one of several methods, yielding a value of the output variable.

5.2.3 Hidden Markov Models

The hidden Markov Model is an unsupervised or semi-supervised method for classifying time series data. It is based on the Markov assumption: the assumption that the system's current state is only dependent on the current observations of the system and the previous state. The states are "Hidden" and the only observations that can be made are the emmisions, which in turn are modelled as being dependent on the current state. Four sets of values form the parameters of the model: 1. Initial probabilities, 2. Transition probabilities, 3. Emission probabilities and 4. the number of states. These probabilities can be acquired from training data. The HMM has a few distinct strengths when used with the type of data in this thesis:

- The HMM is a Bayesian model and is adjustable by expert inputs.
- The HMM can be used as a unsupervised classifier reducing the need for data annotating.
- The HMM is computationally efficient.
- The HMM Takes time-dependencies into account through the state transition matrix

However, the HMM has the disadvantage of not being suitable for fully supervised machine learning in this case. Additionally, few HMM software packages have support for continuous multivariate emissions.

5.2.4 Random Forests

Random forests are a type of ensemble learning technique that incorporates the combined classifications of multiple trained decision trees as its output. Each tree uses a random subset of features sampled with the same distribution for all trees.

The original Random Forest Fortran code by Breiman et al. ported to R [68] was used as an off-the-shelf algorithm.

The random forest algorithm has the additional advantage of being able to estimate variable importance in classification upon completion of training.

5.2.5 R caret package

The R caret package available from the CRAN network provides a easy-to-use and intuitive system for data cleaning, pre-processing, training, classification and evaluation. The Random Forest, Neural Network and Boosting models in caret were used.

5.3 Training label formulation

Devising accurate labels, describing the events observed in the aggregated data, is essential for training and testing the machine learning algorithms in this chapter. The aim of training label formulation is to generate a single, most probable label for each discrete step in time. This time step size is 0.5 seconds, as chosen in Section 4.5.1. MEMS sensor data is not used to assign training and testing labels in order to keep the

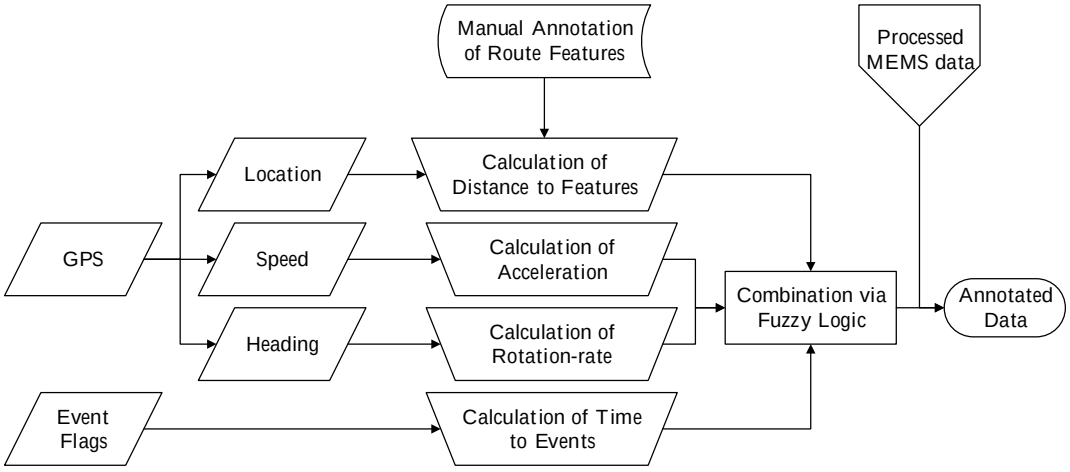

Figure 5.1: An information-flow diagram of the developed annotation algorithm.

training labels and the data on which the machine learning algorithms are trained independent. The information available for classifying the events observed in aggregated data is as follows:

- **GPS speed** Used to detect when the vehicle is stationary. The GPS speed is also differentiated to identify coasting, accelerating and braking.
- **GPS heading** The GPS heading is differentiated to produce a rotation rate, used to identify turns. Due to the slow update rate of GPS data, this rotation rate cannot be used to identify brief events such as swerves or lane changes.
- **GPS coordinates** The GPS coordinates are used to correlate events with manually-annotated physical elements on the route, e.g. turns and speedbumps. Figure 5.2 shows the marked features on the map. Proximity to these marked features was used to identify events.
- **Flagging data** In Section 3.4.3 the flagging of various events by users is described in detail. These flags describe, among other things, the type of driving event that was initialised at a specific moment in time. The time proximity of the flag describing a specific event is used as to identify all flagged events.

This data, used to derive training labels, is very definitive in its description of events. The different types of data do, however, overlap in the events they describe. A turn can, for example, be identified from a high rotation rate, time proximity to a turn flag or geometric proximity to a physical turn on the route. In order to combine these variables in a logical way with the ability to vary the weights of the different descriptive variables, fuzzy logic (described in Section 5.2.2) was used.

The universe of discourse was initialised between 0 and 20 with a granularity of 0.01, since this interval required the least amount of modification of the existing data. Data was appropriately scaled, shifted and limited to fit the interval. GPS speed in $m.s^{-}1$, time to flags in seconds and distance to physical route segments in meters were kept in their original units of measurement. The absolute value of double the rotation rate

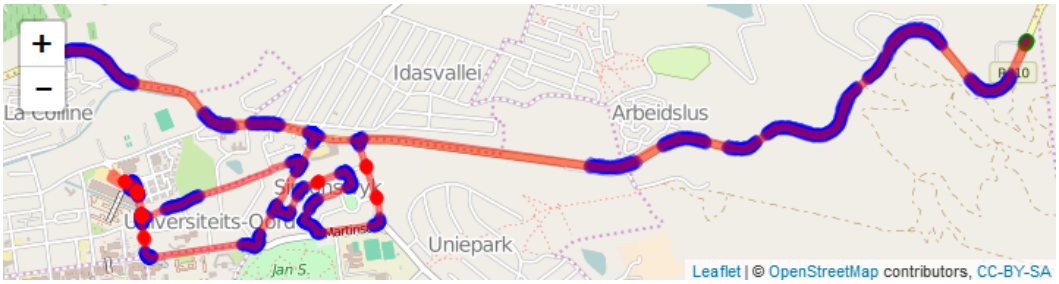

Figure 5.2: A map of the flagging test route with physical turns, U-turns and speed bumps indicated.

was used as input to better fit the interval and to handle left and right turns identically.

For each data type, linguistic variables were defined using normal distributions with chosen standard deviations and means to specify degree of membership. A linguistic output variable was defined for each label type, also using normal distributions for the degree of membership. Figure 5.3 shows the degree of membership for one linguistic variable. The fuzzy rules for each output variable were then defined. The definitions of the fuzzy variables and rules used are given in Tables A.1 and A.2 in Appendix A.

Fuzzy inference and defuzzification are performed on each input data time-step in order to calculate a degree of membership w.r.t. each event descriptive label. Comparing and finding the maximum degree of membership for each point in time yields the most likely label for the time-step in question.
The resulting labels were plotted using the map function of the web app developed in Section 3.7. The plotted data could then be reviewed in detail and compared alongside additional relevant data. Iterations were made to adjust the degree of membership functions for the output variables in order to increase weight of specific labels and thereby improve accuracy of classification.

5.3.1 Fuzzy Set

The Fuzzy rules, defined for each label type in Table A.2 (Appendix A) form part of the expert input to the fuzzy logic system. The units of the linguistic input variables, described in Table A.1 (Appendix A), are chosen to ensure that the useful range of the input variables fall within the chosen universe of discourse (0-20).

5.4 Pre-Processing and splitting of data

Minimal pre-processing was attempted on the data. Performing PCA on the data reduced the accuracy of the results significantly. This can be attributed to the fact that the data was already chosen as a very information rich set of independent dimensions, therefore, variance along the first 5 principal axes were not significantly higher than the variance along the last three principal axes. Due to the time series nature of the data, the splitting into testing and training datasets had to be done in a way that would preserve the temporal dependencies of the data. The data was eventually split by driver. Of the 16 drivers a randomly selected group of 12 drivers' sessions were chosen as the training set and 4 drivers' sessions selected as the testing set.

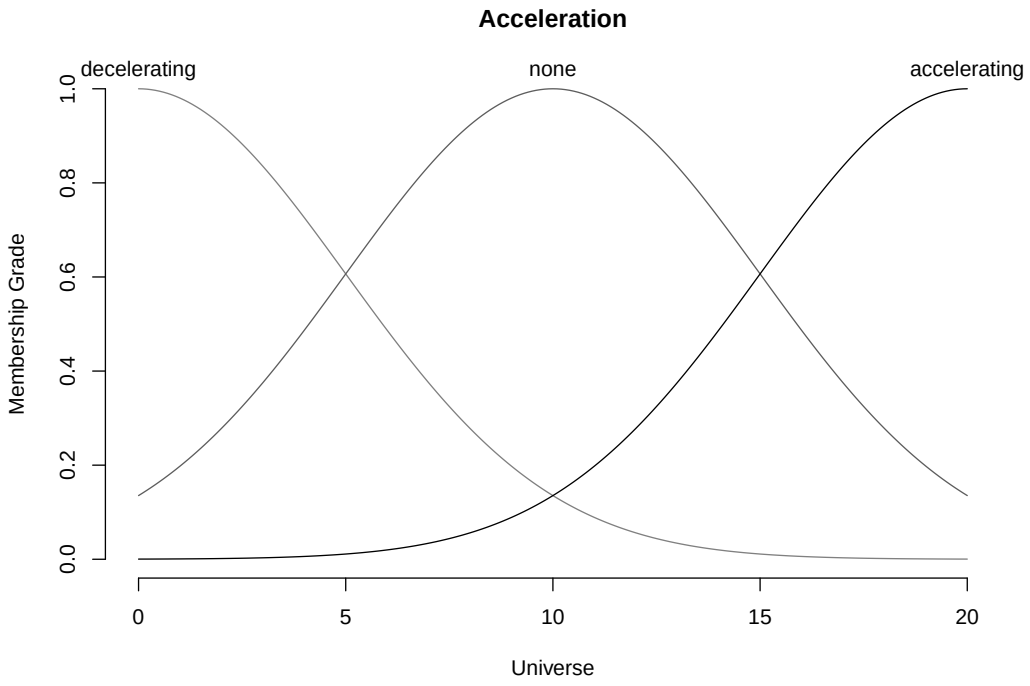

Figure 5.3: Plot showing the normal distributions describing the degree of membership of the *Acceleration* linguistic variable.

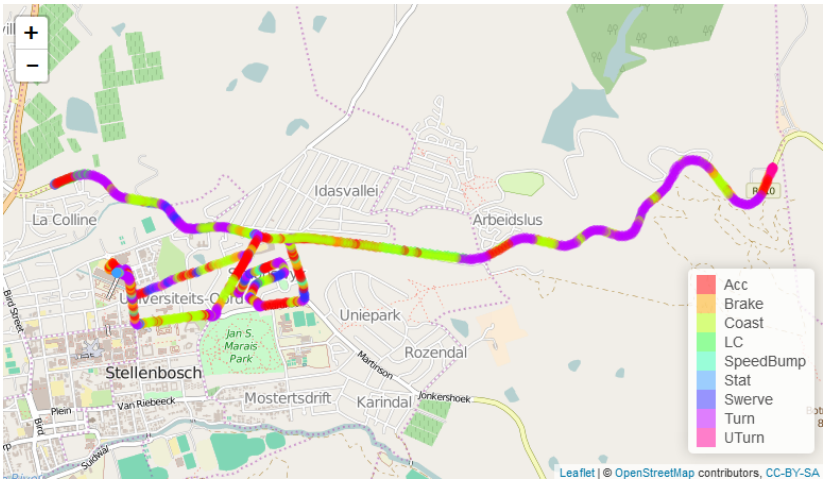

Figure 5.4: A Plot of one set of training data labels.

5.4.1 Windowing

A rolling window of 10 samples with 9 sample overlap is concatenated and used as a single sample for training. The label of the window was chosen as the label corresponding to sample number 5 in the window. This method is useful if there is a constant or slightly varying time delay (or advance) between the data and the label for the data as the machine learning techniques are able to disregard variables of less importance. One disadvantage of this windowing method is the size of the data vector produced and subsequently used for machine learning. The input vector will be 100 dimensional (10 dimensions per sample and 10 samples long) and due to the curse of dimensionality [69] Euclidean distance measures will be meaningless, eliminating machine learning methods such as K-means clustering. The size of the vector also affects computational complexity eliminating some computationally expensive algorithms such as Deep Neural Networks. A high dimensional data vector with a comparatively small number of training samples is also prone to over-fitting when used with certain machine learning models. This narrows the list of feasible algorithms down significantly and hence Random Forests, Boosting and Neural Networks are applied to the data.

5.5 Hidden Markov Models

The HMM was the first model trained and tested with the dataset produced in this thesis. It was initially used unsupervised with multivariate Gaussian emissions for each state and randomised initial parameters. The results are given in Section 6.3.1, page 81. Although the system did find meaningful separation of states, the results were only promising for the more easily discernible events with larger amounts of training data, such as turning and stationarity.
The next step was the semi- supervised execution. The starting values for the initial state probability and state transition probabilities were calculated from the labelled training data set. The parameters for the multivariate Gaussian emmisions were, however, not calculated due to the lack of support in the chosen software library for fitting the Gaussian and the difficulty of manually describing 90 emission probabilities (10 variables for each of the 9 states).

The HMM was chosen, in part, due to its Bayesian nature and the ability to incorporate expert tuning into the model's parameters. The fact that there are 9 states to be detected and in excess of 10 dimensions per state, the resulting 90 Gaussian models to be tuned, were too many to realistically attempt, and prevented any further expert input.
The underwhelming results thus obtained from the HMM can therefore be attributed mainly to the lack of supervised training of - and expert input to the multivariate Gaussian emission probabilities.

5.6 Random Forest

The training of the Random forest model involved very little user input. The only two parameters adjusted by the user was the "mtry" and "ntrees" parameters. The number of trees was increased logarithmically until the point at which the algorithm was still computationally feasible, the final value used was: "ntree = 1000". The training was also repeated for mtry = $[10, 20, 40, 60]$ and thereafter the optimal value

was iteratively chosen as 45. The Random Forest machine learning algorithm was initially implemented on single time samples of the time series data. The results of the training and testing showed unacceptably low correlation between the input data and the labels to by obtained, especially for brief events. The technique described in Section 5.4.1 was therefore implemented as an attempt to mitigate any time shift between the labels and the data. The windowing method produced the best results, shown in Section 6.3.2 on page 83. The trained Random Forest model is easily used to make predictions given a set of testing data as it is merely a set of binary decisions based on the input data and the rules specified by the trained model.

5.6.1 Variable importance

The variable importance plot in Figure 5.5 shows that "Gyroscope3.6", or the 6th sample of the Gyroscope z-axis in the window, is the most importantly regarded variable. This means that the most important information for classifying a window, of which sample 5 gives the window its label, is sample 6, indicating a half second advance of the annotation w.r.t. the data. Furthermore, the variable importance plot shows that the Gyroscope z-axis or yaw rate is by a large margin the most important variable for classification, followed by the 4 to 7 Hz road vibrations and the cumulative rotation. The accelerometer data is noticeably lacking with only the accelerometer z-axis featuring as the 16th most important variable.

5.7 Summary

To summarise, this chapter is concerned with the process of applying the data generated in Chapter 4 to appropriate event classification techniques. A brief overview is given of the literature concerning the methods employed in this thesis and the rest of the field for classifying multivariate continuous time series data. The methods discussed include the classification methods for varying length segmented time series data, Fuzzy Logic, Hidden Markov Models, Random Forests and methods found in the R caret package. The formulation of training and testing labels for the data is discussed in detail in Section 5.3, where the automated annotation algorithm's development is presented. All pre-processing required for optimal dataset and classification algorithm compatibility is discussed in Section 5.4 before the implementation of the classification algorithms is shown in sections 5.5 and 5.6. The variable importance chart generated by the Random Forest algorithm during training is also analysed. The work done in this chapter produces, directly, the results in sections 6.3 and 6.4 of the following chapter.

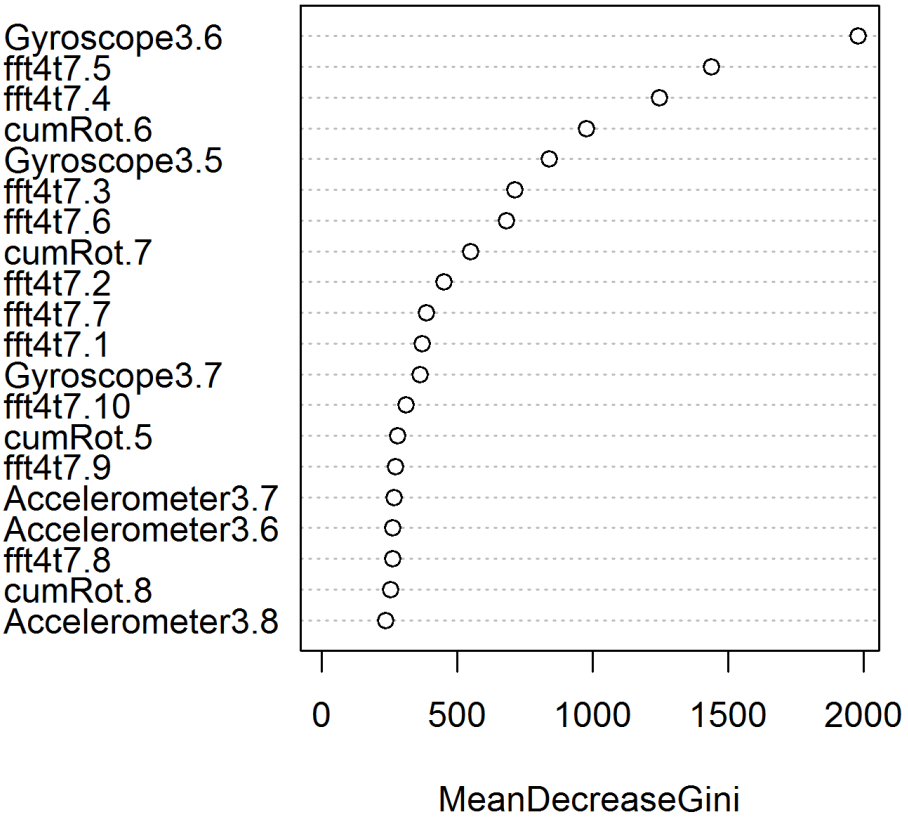

Figure 5.5: A plot of the variable importance of the 20 most important variables, in decreasing order, generated by the random forest algorithm. The number after the period is the index of the variable in the window. The number before the period indicates the dimension of the variable with 1,2 and 3 representing x,y and z respectively.

Chapter 6

Results

6.1 Introduction

In addition to the iterative testing and improvement of the various functional components of the system described in previous chapters, this chapter contains visual as well as quantitative results comparing the outputs of the system components with ground truth measurements. Two main components' outputs are evaluated:

- The reorientation algorithm's output, IMU data transformed to the vehicle axes, is compared to the expected output using the varying orientation test data from Section 3.4.2.
- The classification algorithms' outputs, in the form of individual manoeuvre classifications, are evaluated w.r.t. the annotations from Section 5.3 to the data collected in Section 3.4.3. The best performing machine learning algorithm is analysed w.r.t. its performance in detecting each event type.

The results in this chapter serve as an indication of the functionality of untested components that contribute to the observed outputs, proof of the complete system's feasibility as well as a platform from which to identify and critique components that can be improved with future work. Figure 6.1 shows the different icons used to indicate user assigned flags on the map-extracts.

6.2 Reorientation algorithm

The reorientation algorithm enables the use of orientation dependent MEMS IMU data without introducing additional constraints on smartphone position during use. The data is therefore evaluated by comparing its output with data from a phone constrained to a fixed position and aligned to coincide with vehicle axes and therefore requires no

Figure 6.1: A figure showing all the icons used to indicate user-flags on the map extracts.

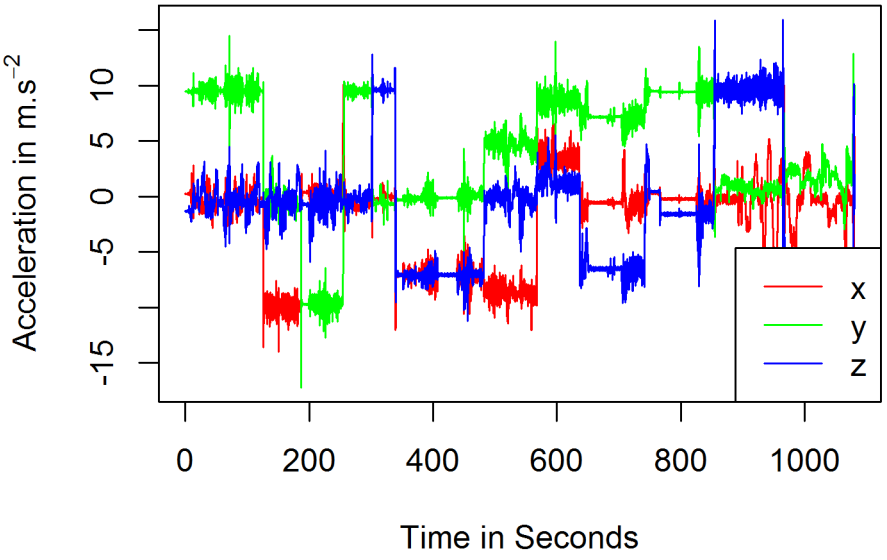

Figure 6.2: A plot showing the accelerometer data recorded by the device with sporadically changing orientation w.r.t. the vehicle.

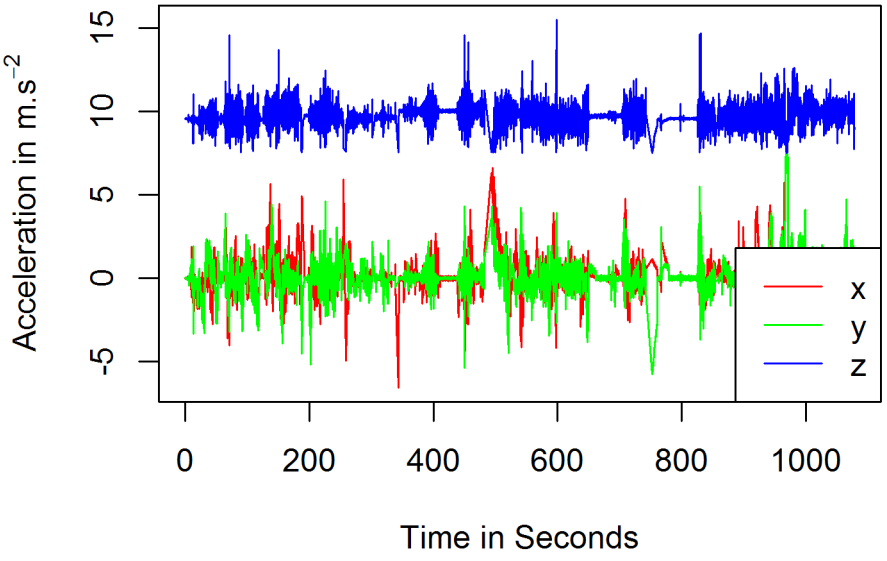

Figure 6.3: A plot of the estimated 3-axis acceleration in the vehicle's frame of reference, excluding moments where the device is detected to be rotating w.r.t. the vehicle.

reorientation. During the recording process, the phone was rotated a total of 15 times. Figure 6.2 shows the raw recorded acceleration data. The changes in orientation can be observed from the change in the measurement of the constant gravity vector.

The output of the reorientation algorithm is a rotated 3-axis rotation - and 3-axis acceleration estimation. The resulting accelerometer vector, excluding the 14 seconds where the device was detected to be rotating is shown in Figure 6.3.

Quantitative evaluation was done by using the RMS error between the data from the stationary device and the rotated estimation after a time shift to accommodate device time differences.

It is clear from tables 6.1 and 6.2 that despite the accuracy of the z-axis, the x-

Table 6.1: A summary of the error-magnitude in the rotated accelerometer data as well as the RMS error when no operation is applied for comparison.

	Accelerometer X	Accelerometer Y	Accelerometer Z
Min.	0.000072	0.000015	0.000001
1st Qu.	0.236391	0.212383	0.143603
Median	0.549811	0.454656	0.318945
Mean	0.937399	0.819486	0.472499
3rd Qu.	1.245621	1.079924	0.654412
Max.	7.966678	10.604249	7.540233
Mean no Op.	3.359860	4.665455	4.414912

Table 6.2: A summary of the error-magnitude in the rotated gyroscope data as well as the RMS error when no operation is applied for comparison.

	Gyroscope X	Gyroscope Y	Gyroscope Z
Min.	0.0000004	0.0000001	0.0000003
1st Qu.	0.0036933	0.0026988	0.0014559
Median	0.0112938	0.0103271	0.0039796
Mean	0.0208388	0.0191751	0.0086501
3rd Qu.	0.0270758	0.0248172	0.0102478
Max.	0.8990581	0.8357807	0.6248296
Mean no Op.	0.0258631	0.0289000	0.0538099

and y-axes perform significantly worse. This can be explained from Figure 6.4.

The inaccurate z-axis rotation in Figure 6.4 took place when the non-stationary recording device was deliberately rotated slowly to a new position to avoid triggering the "Rotating" state. This could be mitigated by implementing an algorithm that continuously changes the z-axis rotation quaternion so the forward vector matches accelerations that lack a corresponding z-axis gyroscope rotation and the forward vector is orthogonal to accelerations that have a corresponding z-axis gyroscope rotation. In so doing, the forward vector can be updated even when no significant turns take place and the vehicle is not stationary.

6.3 Machine learning

The two machine learning methods were evaluated using 25% of the annotated data as a testing set. The results are shown here in the form of truth tables.

6.3.1 HMM

As shown in Table 6.3, the semi-supervised HMM (Initial Probabilities and State Transition Probabilities were calculated from training data) did find some separation of major classes (Turn, Coast, Stationary, Acc, Brake) in the data, however, on average, the model fit the data quite poorly. A fully supervised implementation was attempted, but estimating the 90 emission probabilities (10 variables for each of the 9 states) proved ineffective.

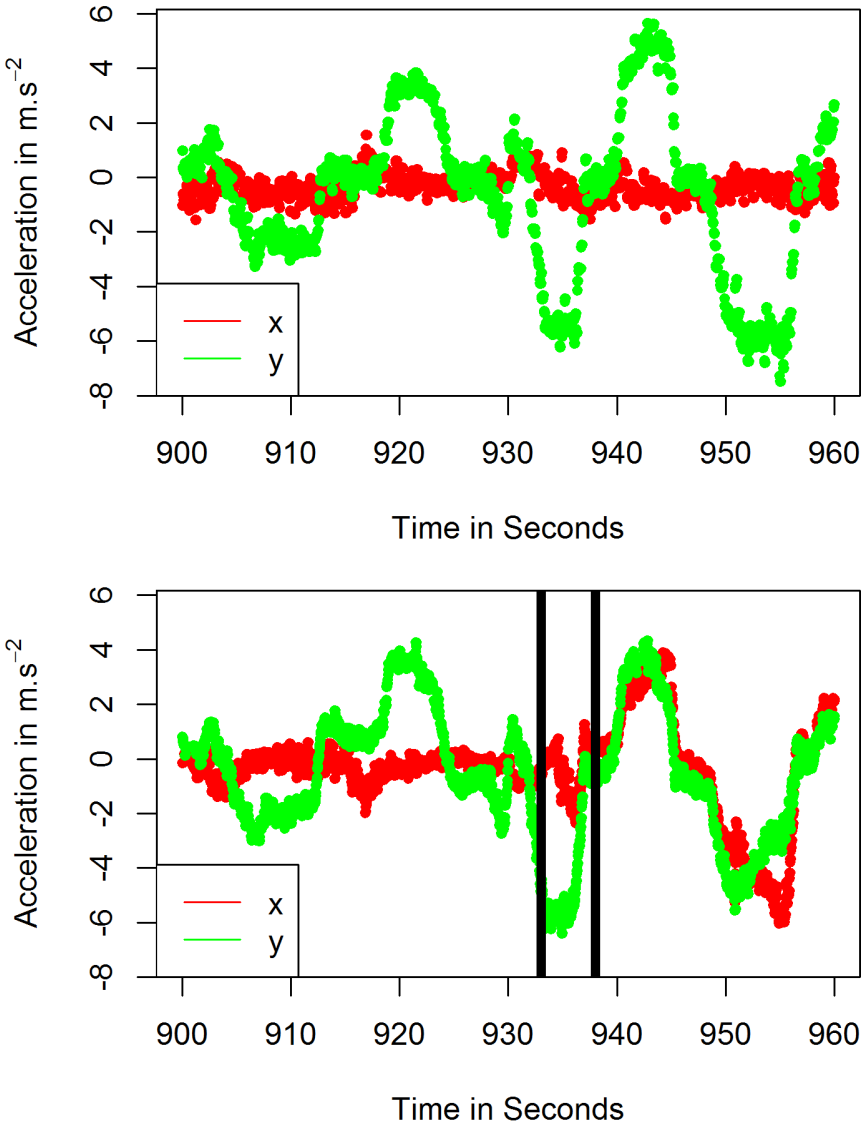

Figure 6.4: A figure showing the accelerometer x- and y-axes ground truth data (top) and the rotated data (bottom) for a 1 minute period . From 900 to 930 seconds, the reorientation algorithm correctly aligned the axes. Between the lines at 934 and 938 seconds however, the device was slowly rotated around its z-axis, this rotation was not detected and the algorithm only reoriented on a sharp turn 30 seconds later.

Table 6.3: A truth table for the HMM classified data, showing the fraction of data from a reference label assigned a corresponding prediction label.

	Reference								
Prediction	Acc	Brake	Coast	LC	Speed Bump	Stat	Swerve	Turn	UTurn
Acc	307	73	754	99	15	8	23	28	3
Brake	24	3	44	6	3	0	3	368	0
Coast	134	82	102	8	8	107	3	43	5
LC	32	38	102	2	4	267	3	14	3
Speed Bump	87	30	131	8	38	0	10	43	0
Stat	0	0	0	0	0	335	0	0	0
Swerve	118	22	232	71	9	15	19	184	3
Turn	23	5	44	8	0	8	3	383	1
UTurn	55	16	44	4	5	15	4	354	26

Table 6.4: A truth table showing the number of data points from a reference label assigned a corresponding prediction label using the random forest model.

	Reference								
Prediction	Acc	Brake	Coast	LC	Speed Bump	Stat	Swerve	Turn	UTurn
Acc	187	6	45	19	18	7	0	33	7
Brake	12	90	13	0	4	8	0	8	0
Coast	465	112	1186	158	39	20	44	138	2
LC	3	0	1	2	0	0	0	1	0
Speed Bump	3	1	0	0	11	0	0	2	0
Stat	39	39	92	0	3	697	0	52	2
Swerve	0	0	0	0	0	0	0	0	0
Turn	77	24	113	31	9	22	25	1183	3
UTurn	2	0	0	0	0	8	0	0	28

6.3.2 Random forest

Of the three generic supervised machine learning algorithms attempted, the Random forest from the *caret* package in R [70] showed the best performance. The results shown in tables 6.4 and 6.5 indicate that a meaningful separation of classes was found by the trained model. For all classes, bar *Coast*, *Stat* and *Turn* sensitivity ($\frac{\sum TruePositives}{\sum Conditionpositive}$) was low, but the specificity was excellent in all cases but coasting. This is in keeping with the strategy to err on the side of neutrality and indicates a practically useful system as it will cause minimal false positives. It will be shown in Section 6.4 that, due to the decision to detect events on a per-second basis and to label any increase or decrease in GPS speed as an acceleration or braking respectively, the distinctions between acceleration, braking and coasting in the annotated dataset are in many cases negligibly small.

Table 6.5: A table showing results of random forest model classification in terms of Sensitivity, Specificity and other relevant statistical parameters.

	Acc	Brake	Coast	LC	Speed Bump	Stat	Swerve	Turn	UTurn
Balanced Accuracy	60%	66%	78%	50%	57%	93%	50%	88%	83 %
Sensitivity	24%	33%	82%	1%	13%	91%	0%	8%	67%
Specificity	97%	99%	73%	100%	100%	95%	100%	92%	100%
Prevalence	6%	3%	43%	0%	0%	18%	0%	29%	1%
Pos Pred Value	58%	67%	55%	29%	65%	75%	0%	80%	74%
Neg Pred Value	87%	96%	91%	96%	99%	98%	99%	94%	100%
Detection Rate	4%	2%	23%	0%	0%	14%	0%	23%	1%

6.4 Individual event results

In this section each of the manoeuvres for which detection is attempted are analysed in terms of the results of the machine learning as well as the processed data used for the machine learning. All discussions will refer to the detection results in tables 6.4 and 6.5.

6.4.1 Accelerating and braking.

As shown in Table 6.5, the balanced accuracy for detecting an acceleration event is 60% which is low and therefore required further investigation. From the truth table (Table 6.4) it can be deduced that the majority (59 %) of accelerations are classified as *Coasting* by the trained random forest model. It is therefore important to look at the distinction between acceleration and coasting in terms of the variables that would be used by the automated annotation algorithm as well as the random forest model for classification. These variables are: the GPS speed in Figure 6.6, the acceleration flags used in the annotation algorithm (Figure 6.5) and the longitudinal acceleration fed to the random forest model in Figure 6.8. The data displayed in Figure 6.5 is from the time-interval 270 to 290 seconds as are the other figures in this section.

As shown by the map extract in Figure 6.5, the *Brake* event before the speed bump as well as the *Acc* event towards the end of the interval are misclassified as *Coast*. Figures 6.7 and 6.6 show that the annotations are predominantly correct and there are a deceleration and acceleration that occur. In Figure 6.8 only two sharp peaks of acceleration and deceleration are visible, this explains why only the initial accelerations are correctly classified. The loss of the rest of the longitudinal acceleration magnitude could be attributed to the gravity removal algorithm using the Android provided *Gravity* [41]. The Android algorithm could be estimating the direction of the gravity vector in the direction of the acceleration, causing the gravity vector to be subtracted in the incorrect axis. This problem could be mitigated by implementing

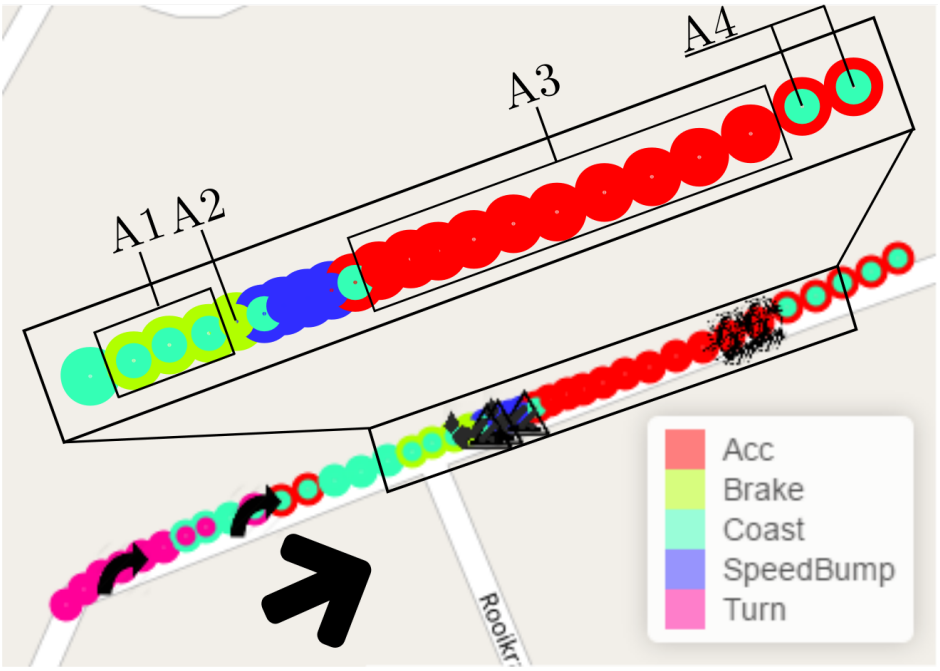

Figure 6.5: A segment of a map from the data visualisation web app with the reference labels as well as the output classification of the random forest algorithm plotted as coloured circles. The inner circle represents the latter mentioned. The flags used to annotate the dataset are also shown as icons on the map. Sections A1 and A4 show misclassified events while sections A2 and A3 show correctly classified events. The vehicle moves in the direction of the arrow.

the algorithm proposed by the author in a paper published for another project titled: *Acceleration estimation using smartphone-based sensors* [62].

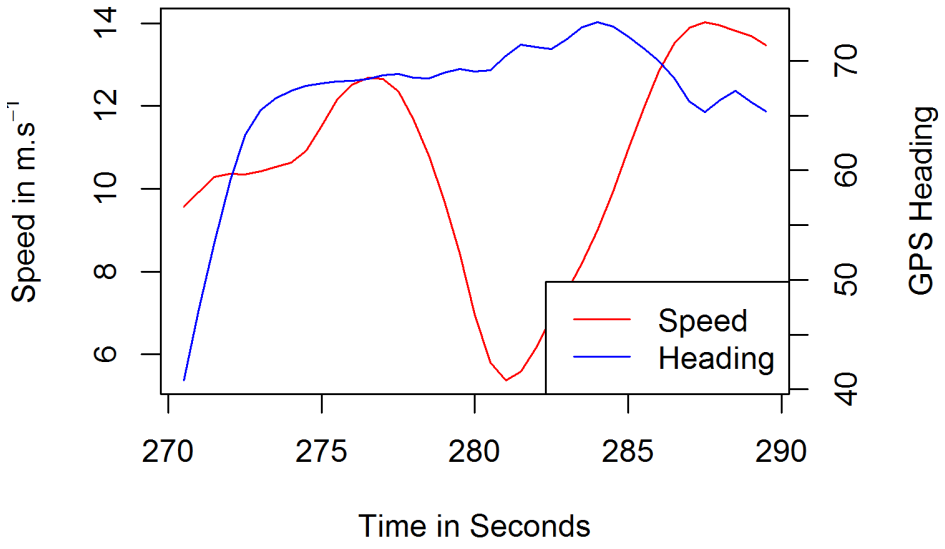

Figure 6.6: A plot of the GPS speed and heading over a chosen, 270 to 290 seconds, interval showing a sequential acceleration and deceleration.

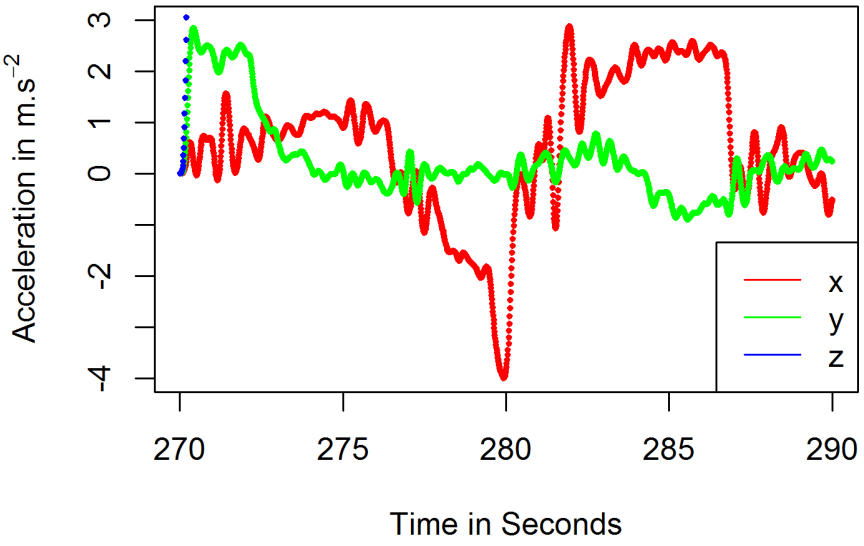

Figure 6.7: A plot of the raw MEMS X- and Y-axis acceleration measurement over the chosen, 270 to 290 seconds, interval showing a sequential acceleration and deceleration. A 10Hz low pass filter was applied to improve legibility.

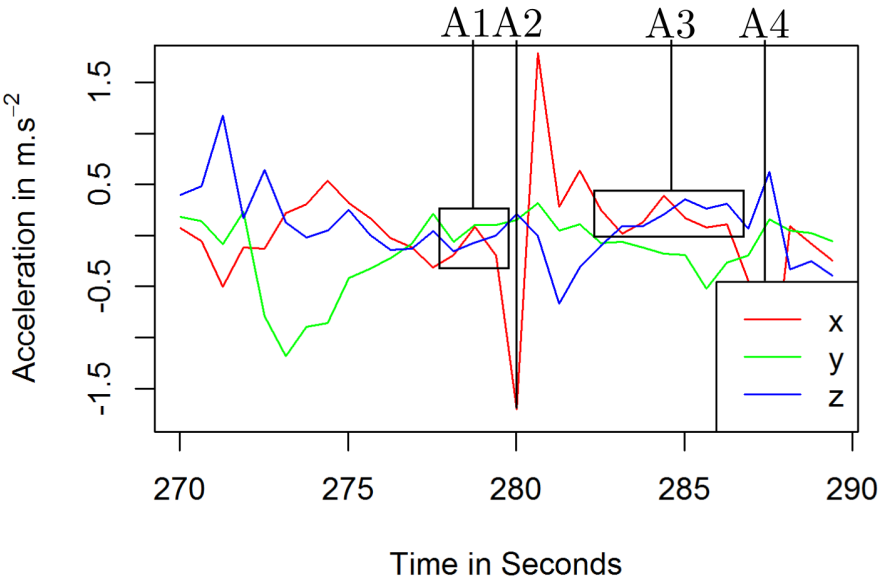

Figure 6.8: A plot of the 2Hz processed acceleration measurement over the chosen, 270 to 290 seconds, interval showing a sequential acceleration and deceleration. Sections A1 and A4 show the data used when events were misclassified while sections A2 and A3 show the data used to correctly classify events. It is clear when this data is compared to Figure 6.7, that there is a loss of acceleration magnitude in the processing, most likely caused by the gravity removal algorithm.

6.4.2 Coasting & Stationary

Coasting and *Stationary* states were both detected with good sensitivity (82% and 91 % respectively) and specificity (73% and 95% respectively). The slightly lower specificity for *Coasting* can be attributed to the smooth transitions between a coasting vehicle and the commencement of an event. Detecting when a vehicle is coasting and stationary,

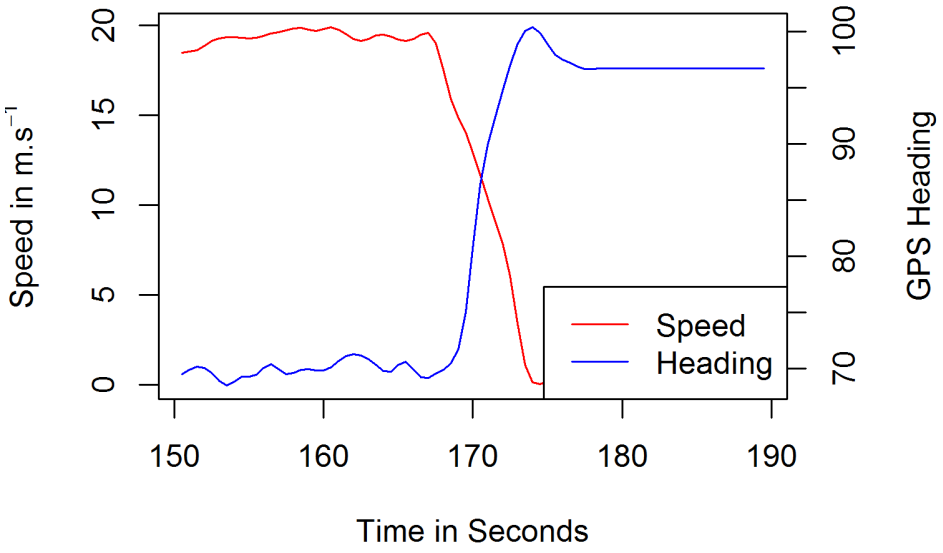

Figure 6.9: A plot of GPS speed and heading over a chosen, 150 to 190 seconds, interval showing a section where the vehicle is coasting and followed by a turn and a section where the vehicle is stationary.

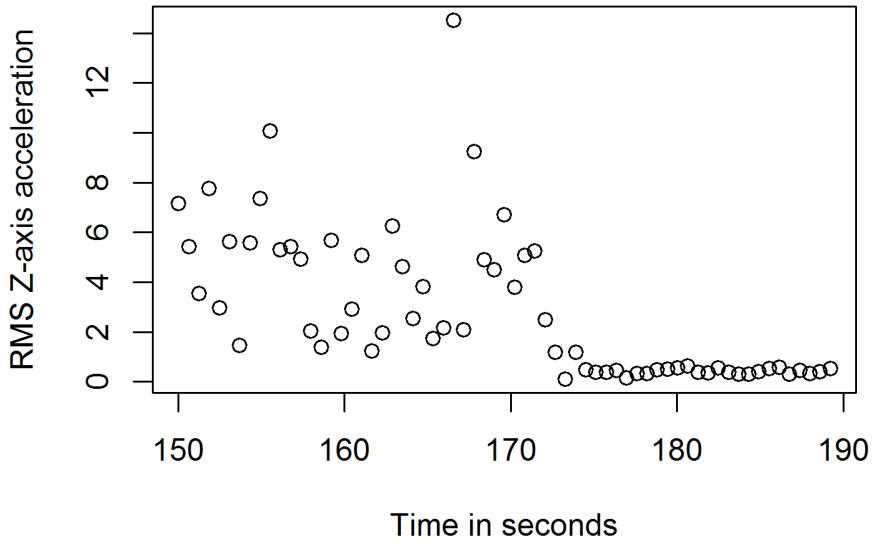

Figure 6.10: A plot of the RMS windowed and bandpass filtered Accelerometer reading over a chosen, 150 to 190 seconds, interval showing the lack of vibrations when the vehicle is stationary.

requires information from the accelerometer and gyroscope in order to determine that no events are taking place as well as the 4 to 7 Hz vertical accelerometer axis signal magnitude to distinguish road-vibrations and, consequently, whether the vehicle is moving. Figure 6.10 shows the clear distinction between the stationary and moving vehicle from the windowed and 4 to 7 Hz bandpass filtered vertical axis acceleration.

6.4.3 Turn and U-Turn

In combination, *Turn* and *U-Turn* have the same prevalence in the dataset (28%) as coasting. The large amount of data has enabled the model to detect these with high accuracy (Balanced accuracies of 88% and 83% respectively). The lower sensitivity of the *U-Turn* classification (67%) can be explained by looking at the false negative predictions, of which 3 (7%) are misclassified as *Turn* and 7 (17%) as *Acc*. This indicates a two-fold problem. Two of the 3 erroneously classified as *Turn*(T1) and 3 of the 7 misclassified as *Acc*(T2) are shown in Figure 6.11 as purple and red dots on pink respectively. Looking at Figure 6.12, it is clear that the driver "oversteered" after the U-Turn and this minor turn in an opposing direction to that of the U-turn was classified by the algorithm as a separate turn.

At the time of the *Acc* classifications, it is clear from the route geometry that the U-turn has been completed, and the vehicle is accelerating, it is however a side effect of the automated annotation algorithm that the physical location of the U-turns are specified as a generalised attribute of the route and the same geometry was used for all tests. The geometry defined is shown in Figure 5.2 on page 74. An iterative adjustment of the manually defined U-turn location can therefore by made to improve the reference label accuracy on the fringes of the event.

A similar effect can be observed with standard turns. Misclassifications are often limited to the edges of events as can be seen in Figure 6.11. More lenient definitions of event boundaries would not decrease the system's utility, but will make for more impressive figures.

The classifications for Turns and U-turns are therefore significantly more accurate than even the reference labels.

6.4.4 Speed Bump

Classification of the traverse of a Speed Bump was only performed with a balanced accuracy of 57% despite it being a very observable event when using MEMS sensor data (as can be seen in Figures 6.13 and 6.14). Twofold reason for the misclassification was identified: Firstly, the event boundaries come into play again. Figure 6.15 shows that the first (lower) speed bump has 6 samples with *Speed Bump* reference labels and only 2 with *Speed Bump* classifications (only 33%, despite a successful detection). Similarly, the second(upper) speed bump has 9 samples with *Speed Bump* reference labels and only 1 with *Speed Bump* classifications (only 11%, despite a successful detection). This is, once again, a drawback of the automated annotation algorithm using a combination of inaccurate user *Flag* location and too-wide geofence definitions. The second reason identified for the ineffective speed bump sensitivity result is the lack of speed bump data. The speed bump data only makes up 1.6% of the training and testing data sets, even with the widely defined boundaries .
Increasing the amount of speed bumps on the route while decreasing the geofence boundary size for the automated labelling algorithm would significantly improve the sensitivity figures.

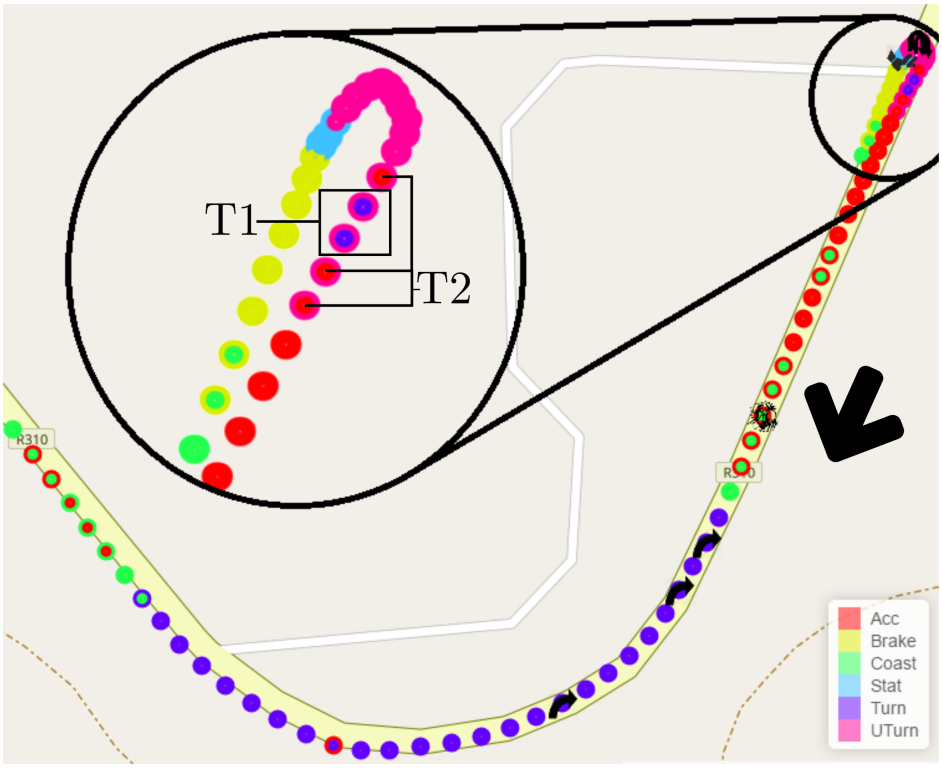

Figure 6.11: A segment of a map from the data visualisation web app with the reference labels as well as the output classification of the random forest algorithm plotted as coloured circles. The inner circle represents the latter mentioned. The flags used to annotate the dataset are also shown as icons on the map. T1 represents the oversteer correction that was incorrectly annotated as a U-turn and classified as a turn. T2 points to samples incorrectly annotated as part of the U-turn and classified as Accelerations. The vehicle performs a U-turn and then moves in the direction of the arrow.

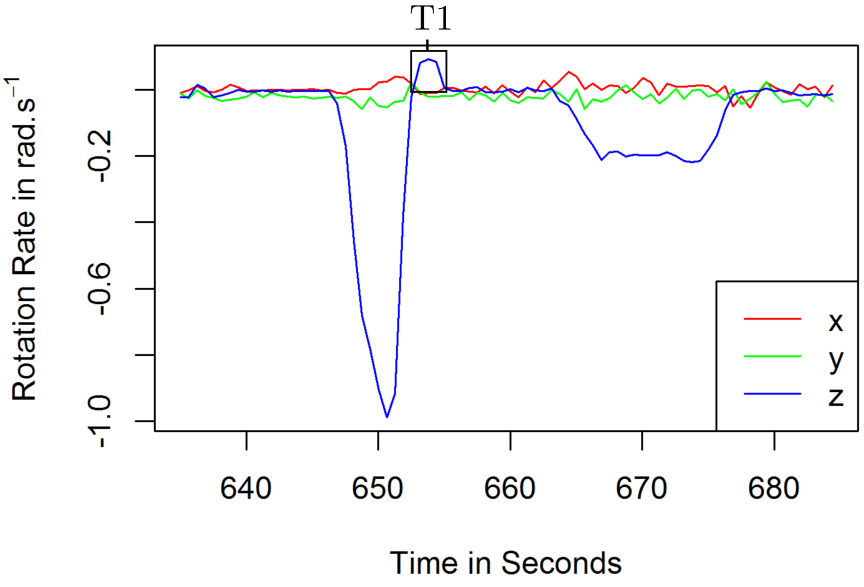

Figure 6.12: A plot of the processed 2Hz gyroscope measurement over the chosen, 635 to 685 seconds, interval showing a sequential U-turn, oversteer correction turn and prolonged turn. T1 points to an oversteer-correction turn after the U-turn.

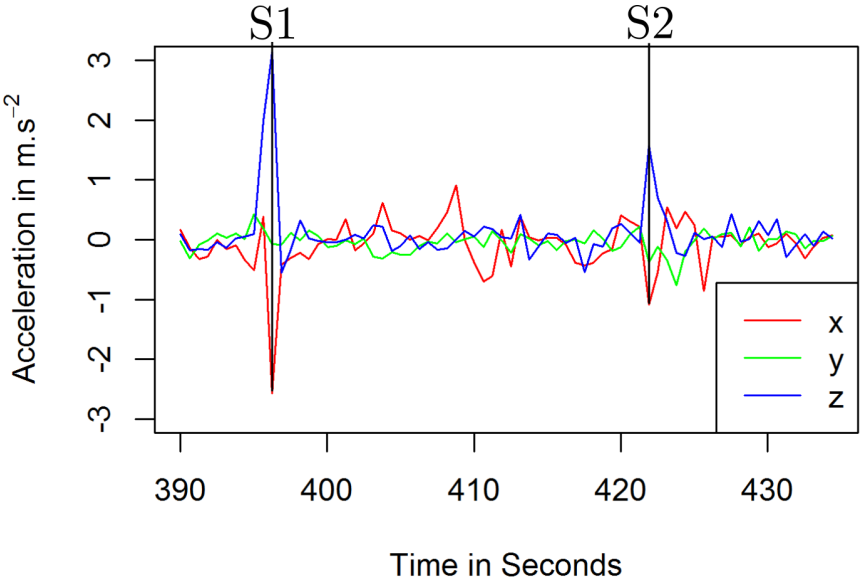

Figure 6.13: A plot of the processed 2Hz accelerometer measurement over the chosen, 390 to 430 seconds, interval showing two consecutive *Speed Bump* events (S1 and S2).

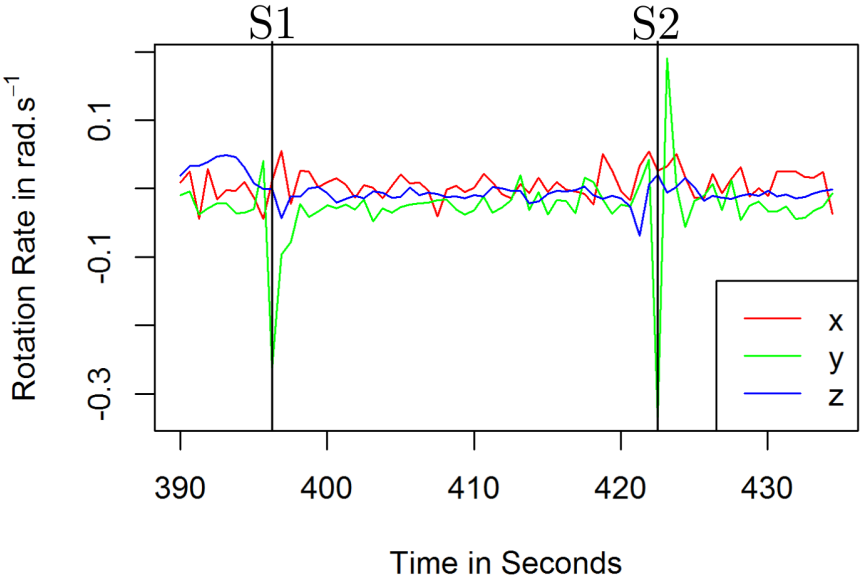

Figure 6.14: A plot of the processed 2Hz gyroscope measurement over the chosen, 390 to 430 seconds, interval showing two consecutive *Speed Bump* events (S1 and S2).

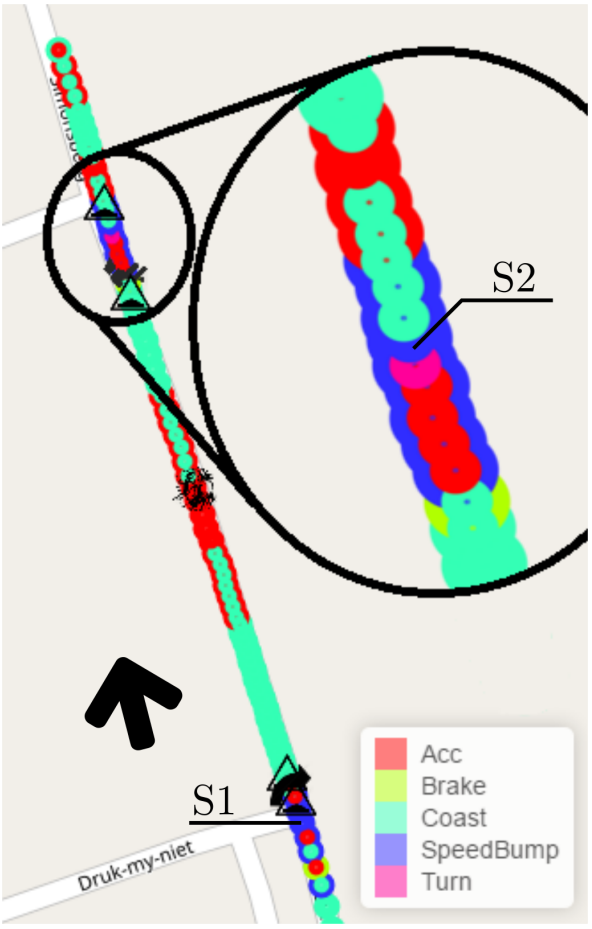

Figure 6.15: A segment of a map from the data visualisation web app with the reference labels as well as the output classification of the random forest algorithm plotted as coloured circles. The inner circle represents the latter mentioned. The flags used to annotate the dataset are also shown as icons on the map. The vehicle moves from the bottom to the top of the figure. S1 and S2 represent the moment of impact on the two speed bumps.

6.4.5 Swerve and Lane Change

The *Swerve* and *Lane Change* events showed the worst detection rates, suffering from the same two problems of poorly defined event boundaries in the annotations and lack of sufficient samples for the training of the machine learning algorithm. These two events, however, suffer the additional problem of not having geographically stationary locations that can contribute to defining their positions in the reference set. The annotation of these events are done solely based on user *flags* as can be seen in Table A.2.

The minimal changes in location, speed and heading during these events eliminate the use of GPS as a viable method for improving the accuracy of the annotation locations (as proved in Section 3.3). Using the existing dataset, the annotations can be improved by manually seeking out the disturbances in the MEMS sensor data near user *Swerve* and *Lane Change* flags.

6.5 Summary

In summary, it was proven that the reorientation algorithm functioned for most recalibration scenarios, but could be fooled by slowly rotating the device around the vehicle's vertical axis while the vehicle is moving, so its rotation characteristics are similar to that of the vehicle. A method for mitigating errors of rotation around the vehicle Z-axis is proposed, but not implemented or tested. A HMM machine learning algorithm was implemented, but the resulting model did not perform well with the data. Three additional machine learning algorithms were implemented: Boosting, Neural Networks and Random Forest. The Random Forest algorithm performed best and it's performance was analysed in detail with respect to each event.

All classes' sensitivity rates were adversely affected by inaccurate annotations to the dataset. In many cases shown in this chapter, the machine learning classification of the data was more accurate than that of the automated annotation algorithm. This was proven by extracting representative scenarios from the data and showing from the unprocessed data that the machine learning classification is correct. Harsh acceleration and braking events were detected with high accuracy, but, due in part to Android's "Gravity" sensor being influenced by prolonged accelerations and decelerations, these longer accelerations and decelerations were filtered out by the gravity removal algorithm. This resulted in the detection accuracy for these events being reduced. *Coasting*, *Stationary* and *Turn* events were detected with high accuracy. *U-turn*, *Speed Bump*, *Swerve* and *Lane Change* events were under-represented in the dataset, but *U-turn* and *Speed Bumps* events were still detected with acceptable accuracy. Improved annotation of the *Swerve* and *Lane Change* events in the dataset is needed to facilitate improved model training.

It is concluded that the results in tables 6.4 and 6.5 are not representative of the accuracy of the Random Forest classifications. All events, apart from *Swerve* and *Lane Change* were detected adequately.

Chapter 7

Conclusion

The field of vehicle telematics has been revolutionised by the resurgence of mobile- and sensor technology. As mobile phones continue to increase in pervasiveness, processing capacity, connectivity and richness of sensors, more and more dedicated hardware systems designed for vehicle telematics will be replaced by the widely obtainable and affordable smartphone. Using a smartphone as a sensing platform however, does have its significant challenges. Firstly, the smartphone is completely under the control of the end-user. Anything an application wants to access, including sensors and GPS, requires the user's permission and it is always within the user's (or even the Android operating system's) capacity to end the application life cycle if it causes any inconvenience, e.g. using battery-hungry sensors or requiring constraints on device position during application operation.

Most smartphone-based vehicle monitoring systems are directly dependent on GPS for information on vehicle kinematics. Using smartphone-based GPS continuously has the side-effect of drastically reducing phone battery life (see page 33). Additionally, the kinematic information provided by GPS is often of limited accuracy, with low sampling rates (1Hz) and the possibility of signal loss and slow signal acquisition times, as shown in Section 3.3.

The drawbacks of a GPS-based systems has driven research into alternative methods for detection, especially the fast evolving MEMS sensors. These sensors, when used from a smartphone-based platform, have their own notable challenges that need to be mitigated. With regards to minimising inconvenience: The problem of sampling from a moving device located within a moving vehicle, without setting stringent constraints on device position, needs to be addressed. With regards to feasibility: 1. Sampling from a non-real-time operating system creates problems with unevenly spaced data samples. 2. Smartphones have few generalised hardware specifications and no specifications as to the quality of the MEMS sensors used, the quality of sensors between devices can vary widely. 3. Despite the constant improvement of mobile processing power, mobile computing power is limited and comes at a premium, as it is also directly related to smartphone battery drain.

This led to the hypothesis being stated: A generalised sensing platform can be developed to identify and describe driving events by exclusively using MEMS sensors in an unmounted smartphone.

The research objectives were chosen to, in combination, prove this hypothesis beyond doubt.

7.1 Summary of research objectives

The research question was addressed systematically by breaking the problem down to a set of 7 research objectives that can verify the 5 dissertation statements in Section 1.3. Figure 1.1 on page 5 gives an overview of the connections between the dissertation statements and the sections where they are addressed. The research objectives and their fulfilment are discussed in this section with periodic reference to statements verified by the objectives.

7.1.1 To develop a complete smartphone-based system for collecting and annotating driving data.

This objective holds in itself a set of two challenges that need to be overcome in order to regard it as fulfilled:

7.1.1.1 Driving events

The annotations used for "...annotating driving data." must reflect what is to be detected using the data: driving events. Identifying driving events that are of relevance depends greatly on the specific application, however, a general set of driving events deemed noteworthy, and that form a fully descriptive set for day-to-day driving were chosen in Section 3.2. The chosen events are:

- Acceleration
- Braking
- Lane Changing
- Swerving
- Speed Bumps
- Turns
- U-turns
- Rough Road Surfaces
- Coasting
- Stationarity

The Rough Road Surfaces event type was dropped due to its subjective nature and the difficulty of defining boundaries to the event.
The events were chosen based on what is deemed important to detect in the literature.

7.1.1.2 Application

The "...smartphone-based system..." part is the crux of the objective. Two applications had to be developed to facilitate data collection and annotation: 1. A logging application, that records driving data according to specifications. 2. A flagging application, that enables passengers in a vehicle to enter their observations of driving as flags through an intuitive GUI.

These two apps are designed to synchronise their times to the nearest millisecond from the same Network Time Protocol (NTP) time-server. The development and testing of these tools are described in Section 3.5. Both applications were deemed fully operational and were used for data capture and flagging.

7.1.2 To collect a raw GPS and MEMS sensor dataset with passenger annotations that is representative of general driving events, using smartphones.

In order to collect a "...dataset that is representative of general driving events..." a route for data collection had to be chosen, so it contains all the events of interest in adequate proportions and at different speeds and road gradients. Table 3.4 on page 37 shows the number of events of each type recorded on the chosen route. The chosen route is shown in Figure 3.10. The setup for the test is discussed throughout Section 3.4.3. The chosen driving scenario was representative of general observed driving event. In excess of 10 million data points containing more than 5000 annotated events from 21 drivers were successfully collected, achieving objective 2.

7.1.3 Compare the efficacy of detecting driving events using GPS vs. MEMS sensors in terms of the accuracy and convenience of use.

The choice of sensors is thoroughly examined in Section 3.3, where two popular systems for collecting information regarding vehicle kinematics are compared in terms of their abilities to detect some of the events chosen in Section 3.2. The conclusion is made that MEMS sensors outperform GPS in various aspects of detecting the manoeuvres relevant to this thesis. Despite GPS still having some indispensable functionality with regards to location sensing, its high battery drain makes it less suitable for use in this application. This optimal sensor selection proves dissertation statement 1.

7.1.4 To develop a tool for analysing and visualising collected data.

Raw data from MEMS sensors, GPS coordinate data and quaternion orientation data are difficult to comprehend if not plotted or otherwise visualised regularly. The data visualisation web app was designed to streamline user interaction with the data and thereby save time and effort while opening up broader insights into the collected data. The specifications and development of the system is discussed in detail in Section 3.7. The developed web application was used throughout the thesis among other things to extract frequency domain information (Section 4.5.1) of particular relevance for identifying driving events..

7.1.5 To develop a system that can process raw smartphone MEMS sensor data into a dataset that is relevant to vehicle dynamics, and that is compatible with machine learning methods.

This objective is arguably the most notable and a primary contribution to this thesis. Sections 4.6 and 4.5 describe the majority of the development process and Figure 4.5 on page 60 shows an overview of the functionality of the developed algorithm. During the development process, an effort was made to employ methods in accordance with sound digital signal processing principals and thereby maximise the ratio of information extracted to algorithm complexity. This processing pipeline includes the gravity removal method and the novel reorientation algorithm. The reorientation algorithm uses information related to the axes of most and least variance, cross product of gyroscope and acceleration, presence of road vibrations and other parameters, when available, to rotate data from the device to the vehicle axes. The reorientation algorithm is tested separately and the results are given in Section 6.2. The results of the reorientation algorithm tests proves dissertation statement 2. The proof of the complete system's functionality lies in the evaluation done in sections 6.3 and 6.4. There it was concluded that an accurate machine learning model could be trained and tested on the data produced by the developed data processing pipeline; thereby verifying objective 5 and dissertation statements 3 and 4.

7.1.6 To develop an algorithm that annotates the processed MEMS sensor dataset using collected passenger annotations and GPS data.

A major challenge faced when training a computer to perform classification is obtaining accurate annotations. Driving data often contains multiple events in close succession followed by longer periods of inactivity. In the 26 hours of collected data, only about 5000 events were classified. A study by Schreiner et al. [71] showed that, even with their specifically developed driving annotation tool it takes 1-7 minutes to annotate 1 minute of driving data. Manual annotation of recorded data was therefore avoided in the interest of producing a system that could be expanded to a much larger training set as well as generate labels on a continuous (2Hz) basis.

The design of the algorithm to produce the labels is given in Section 5.3. A fuzzy logic system was implemented to combine the information given by differentiated GPS heading and speed, manually annotated route features and passenger flags and derive a most likely label.

The algorithm was evaluated by manually reviewing the annotations using the data visualisation web app. The manual review indicated adequate annotation accuracy. During the machine learning algorithm evaluations however, it was found that many misclassifications on the boundaries of events could be attributed to differing subjective definitions of event boundaries between the annotation algorithm and the machine learning algorithms. Brief events such as swerves and lane changes were poorly annotated, owing mostly to the GPS update rate being too low to be used to annotate these brief events.

The automated annotation algorithm was predominantly a great success as the labels were accurate enough to train a machine learning algorithm to find a separation

between classes.

7.1.7 Compare the efficacy of suitable machine learning approaches in identifying driving events using the processed MEMS sensor dataset.

Machine learning serves multiple purposes in this thesis. The developed data processing pipeline could not be evaluated in isolation, it had to be tested as it was made to be implemented: with machine learning. The result of the machine learning algorithms therefore directly reflected the efficacy of the systems on which it is dependent, including the data processing pipeline and annotation algorithm. Primarily however, the machine learning classification was an end in itself. Four different machine learning algorithms were trained and tested of which one was the semi-supervised HMM, chosen for its ability to model time-dependency. Two algorithms were ensemble learning methods, one bagging algorithm (Random Forest) and one Boosting. Finally a neural network was also trained. All algorithms were trained with varied parameters. The Random Forest showed the best results, finding a definitive separation between the different classes and additionally producing an estimation of variable importance in performing the predictions. This proves dissertation statements 4 and 5 in addition to proving that objective 7 was reached.

7.2 Deductions

One can deduce from the research done that the developed system, if further effort is expended to create a mobile application, could have great implications for vehicle telematics as well as vehicle information aggregation in general.

7.3 Implications

The completion of this thesis leads to a few implications for the field of smartphone -based vehicle telematics.

1. Given an application that can classify all significant types of driving events, information specific to an event of interest (i.e. gyroscope z-axis for turn, longitudinal acceleration for an accelerating event) can be parametrised and uploaded efficiently to the cloud for additional processing or aggregation.

2. Given an application that can efficiently (without using GPS) classify driving events from any orientation within the vehicle, a device running the application can be left running whenever a vehicle is moving, recording and parametrising all relevant events and uploading data leading up to cataclysmic incidents when they occur. This data can greatly aid the understanding of what happens leading up to vehicle accidents.

3. Given an application that can be conveniently switched on at any time and calibrates itself to its orientation in the vehicle within seconds, this application

can be used by users of public or informal transport to discreetly record the driving behaviour of the driver. Data relating to reckless- or dangerous driving detection can be parametrised and sent to the cloud where a objective rating of the drivers ability or the comfort level of the vehicle can be produced and used to modify ratings for services such as Uber or to inform others of a public or informal transport vehicle's comfort level.

4. Given an application that can efficiently (without using GPS) classify driving events from any orientation within the vehicle, a device running the application can be left running whenever a vehicle is moving. The application can record and parametrise all relevant events and, when events indicate risky driving, the event and associated data can be sent to a Usage Based Insurance (UBI) provider to adjust insurance premiums accordingly.

5. The possibility also exists of extending the system to the domain of activity tracking, where details about driving style could form part of lifestyle analysis data.

7.4 Limitations

The scope of this thesis grew significantly larger as it progressed, therefore, a significant limitation of the thesis is the accuracy of the systems compared to their potential accuracy. Individual parts of the system, such as the reorientation algorithm and the automated annotation algorithm, were only modified up to the points where they performed their required tasks with sufficient accuracy to prove beyond doubt their utility. These algorithms were not, however, iteratively improved beyond this point and further parameter tuning could dramatically improve results.

Regarding the collected data, although the collected datasets were adequate to train the machine learning algorithms to such an extent that they could find an impressive separation of classes, additional data, especially for Speed Bumps, Lane Changes and Swerves, would dramatically improve results.

Though the complete system was designed with consideration for the limitations of a smartphone-based sensing platform, the only smartphone applications developed in this thesis were the logging and flagging applications.

7.5 Future work

Future work in this field would undoubtedly involve the fine-tuning of the reorientation algorithm, gravity removal algorithm and random forest algorithm. The collection of additional data and the publication of the dataset with the data visualisation web application would be of great benefit to the field. The final product and primary objective of future work should be the implementation of this system in the form of a smartphone application.

Appendices

Appendix A

Fuzzy variables and rules

Table A.1: A table showing the linguistic variables used in the Fuzzy system, their corresponding values and units.

Variable	**Level 1**	**Level 2**	**Level 3**
speed	stat	slow	fast
	0 $m.s^{-1}$	5 $m.s^{-1}$	20 $m.s^{-1}$
acceleration	decelerating	none	accelerating
	0 $m.s^{-2}$.10	10 $m.s^{-2}$.10	20 $m.s^{-2}$.10
rot	none	med	high
	0 $deg.s^{-1}$.2	6 $deg.s^{-1}$.2	15 $deg.s^{-1}$.2
distTurn	close	far	
	0 m.5	20 m.5	
distSB	close	far	
	0 m.5	20 m.5	
distUT	close	far	
	3 m.5	12 m.5	
distTurnFlag	close	closer	far
	0 *seconds*	8 *seconds*	20 *seconds*
distSBFlag	close	closer	far
	0 *seconds*	2 *seconds*	20 *seconds*
distUTFlag	close	closer	far
	0 *seconds*	4 *seconds*	20 *seconds*
distSwerveFlag	close	closer	far
	0 *seconds*	7 *seconds*	20 *seconds*
distLCFlag	close	closer	far
	0 *seconds*	7 *seconds*	20 *seconds*
distBrFlag	close	closer	far
	0 *seconds*	8 *seconds*	20 *seconds*
distAcFlag	close	closer	far
	0 *seconds*	8 *seconds*	20 *seconds*
distRoadSurfaceFlag	close	closer	far
	0 *seconds*	8 *seconds*	20 *seconds*

Table A.2: A table showing the fuzzy logic rules used in the fuzzy system.

IF	speed IS stat	THEN	Stationary IS dead
IF	speed IS fast	OR	
	speed IS slow	THEN	Stationary IS moving
IF	rot IS med	OR	
	rot IS high	OR	
	distTurn IS close	OR	
	distTurnFlag IS close	THEN	Turn IS yes
IF	distTurnFlag IS closer	THEN	Turn IS maybe
IF	rot IS none	OR	
	distTurn IS far	OR	
	speed IS stat	OR	
	distUTFlag IS close	OR	
	distUT IS close	OR	
	distUTFlag IS closer	THEN	Turn IS none
IF	distSwerveF IS close	OR	
	distSwerveFlag IS closer	THEN	Swerve IS yes
IF	speed IS stat	OR	
	distSwerveFlag IS far	THEN	Swerve IS none
IF	distLCF IS close	OR	
	distLCFlag IS closer	THEN	LaneChange IS yes
IF	speed IS stat	OR	
	distLCFlag IS far	THEN	LaneChange IS none
IF	distRoadSurfaceFlag IS close	OR	
	distRoadSurfaceFlag IS closer	THEN	RoadSurface IS yes
IF	speed IS stat	OR	
	distRoadSurfaceFlag IS far	THEN	RoadSurface IS none
IF	distUTFlag IS close	OR	
	distUTFlag IS closer	OR	
	distUT IS close	THEN	UTurn IS yes
IF	speed IS stat	OR	
	distUTFlag IS far	OR	
	distUT IS far	THEN	UTurn IS none
IF	distSBFlag IS close	OR	
	distSBF IS closer	OR	
	distSB IS close	THEN	SpeedBump IS yes
IF	speed IS stat	OR	
	distSB IS far	THEN	SpeedBump IS none
IF	rot IS med	OR	
	rot IS high	OR	
	distTurn IS close	OR	
	distTurnFlag IS close	THEN	Turn IS yes
IF	distTurnFlag IS closer	THEN	Turn IS maybe
IF	rot IS none	OR	
	distTurn IS far	OR	
	speed IS stat	OR	
	distUTFlag IS close	OR	
	distUT IS close	OR	
	distUTFlag IS closer	THEN	Turn IS none
IF	acc IS accelerating	OR	
	distAcFlag IS close	OR	
	distAcFlag IS closer	THEN	Acceleration IS yes
IF	speed IS stat	OR	
	acc IS decelerating	THEN	Acceleration IS none

Table A.3: A second table showing more of the fuzzy logic rules used in the fuzzy system.

IF	acc IS decelerating	OR	
	distBrFlag IS close	OR	
	distBrFlag IS closer	THEN	Brake IS yes
IF	speed IS stat	OR	
	acc IS accelerating	OR	
	acc IS none	THEN	Brake IS none
IF	acc IS none	OR	
	rot IS none	THEN	Coasting IS yes
IF	speed IS stat	OR	
	distBrFlag IS close	OR	
	distAcFlag IS close	OR	
	distTurn IS close	OR	
	distSB IS close	OR	
	distLCFlag IS close	OR	
	distSwerveFlag IS close	OR	
	distRoadSurfaceFlag IS close	THEN	Coasting IS no

Bibliography

[1] S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava, "Using mobile phones to determine transportation modes," *ACM Transactions on Sensor Networks (TOSN)*, vol. 6, no. 2, p. 13, 2010.

[2] M. J. Booysen, J. Gilmore, S. Zeadally, and G. J. Van Rooyen, "Machine-to-machine (M2M) communications in vehicular networks," *KSII Transactions on Internet and Information Systems*, vol. 6, no. 2, pp. 529–546, 2012.

[3] "Global status report on road safety: time for action," Geneva, World Health Organization, 2009, [ONLINE] Available: http://whqlibdoc.who.int/publications/2009/9789241563840_eng.pdf. [Accessed 5 August 2016].

[4] "How DriveCam works," DriveCam, 2010, [ONLINE] Available at: http://www.drivecam.com/our-solutions/how-drivecam-works. [Accessed 5 August 13].

[5] "How it works," AutoHabits, 2012, [ONLINE] Available at: http://autohabits.com/how-it-works. [Accessed 5 August 13].

[6] "Fleet safety," FleetMind, 2013, [ONLINE] Available at: http://www.fleetmind.com/fleet-management-products/fleet-safety. [Accessed 5 August 13].

[7] D. A. Johnson and M. M. Trivedi, "Driving style recognition using a smartphone as a sensor platform," in *14th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2011, pp. 1609–1615.

[8] "Car insurance discounts for safer drivers - aviva drive - aviva," http://www.aviva.co.uk/drive/, (Visited on 07/12/2015).

[9] C. Shi, H. J. Lee, J. Kurczal, and A. Lee, "Routine driving infotainment app: Gamification of performance driving," in *Adjunct Proceedings of the 4th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 2012, pp. 181–183.

[10] P. Händel, I. Skog, J. Wahlstrom, F. Bonawiede, R. Welch, J. Ohlsson, and M. Ohlsson, "Insurance telematics: Opportunities and challenges with the smartphone solution," *Intelligent Transportation Systems Magazine, IEEE*, vol. 6, no. 4, pp. 57–70, winter 2014.

[11] J. Engelbrecht, M. J. Booysen, G.-J. van Rooyen, and F. J. Bruwer, "Survey of smartphone-based sensing in vehicles for intelligent transportation system applications," *IET Intelligent Transport Systems*, vol. 9, no. 10, pp. 924–935, 2015.

[12] J. Wahlström, I. Skog, and P. Händel, "Smartphone-based vehicle telematics-a ten-year anniversary," *arXiv preprint arXiv:1611.03618*, 2016.

[13] D. Trossen and D. Pavel, "NORS: An open source platform to facilitate participatory sensing with mobile phones," in *Fourth Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services (MobiQuitous)*. IEEE, 2007, pp. 1–8.

[14] C. Campolo, A. Iera, A. Molinaro, S. Y. Paratore, and G. Ruggeri, "SMaRTCaR: An integrated smartphone-based platform to support traffic management applications," in *First International Workshop on Vehicular Traffic Management for Smart Cities (VTM)*. IEEE, 2012, pp. 1–6.

[15] O. Briante, C. Campolo, A. Iera, A. Molinaro, S. Y. Paratore, G. Ruggeri, and M. J. Booysen, "ITSPhone: An integrated platform for participatory ITS data collection and opportunistic transfer," *IEEE Infocom 2013*, pp. 1420–1421, 2013.

[16] K. Perera and D. Dias, "An intelligent driver guidance tool using location based services," in *1st International Conference on Spatial Data Mining and Geographical Knowledge Services (ICSDM)*. IEEE, 2011, pp. 246–251.

[17] K. Ali, D. Al Yaseen, A. Ejaz, T. Javed, and H. S. Hassanein, "CrowdITS: Crowdsourcing in Intelligent Transportation Systems," in *Wireless Communications and Networking Conference (WCNC)*. IEEE, 2012, pp. 3307–3311.

[18] X. Zhang, H. Gong, Z. Xu, J. Tang, and B. Liu, "Jam eyes: A traffic jam awareness and observation system using mobile phones," *International Journal of Distributed Sensor Networks*, vol. 2012, 2012.

[19] E. Koukoumidis, M. Martonosi, and L. S. Peh, "Leveraging smartphone cameras for collaborative road advisories," *IEEE Transactions on Mobile Computing*, vol. 11, no. 5, pp. 707–723, 2012.

[20] E. Koukoumidis, L. S. Peh, and M. R. Martonosi, "Signalguru: leveraging mobile phones for collaborative traffic signal schedule advisory," in *Proceedings of the 9th international conference on Mobile systems, applications, and services*. ACM, 2011, pp. 127–140.

[21] M. Lan, M. Rofouei, S. Soatto, and M. Sarrafzadeh, "SmartLDWS: A robust and scalable lane departure warning system for the smartphones," in *12th International Conference on Intelligent Transportation Systems (ITSC'09)*. IEEE, 2009, pp. 1–6.

[22] H. Eren, S. Makinist, E. Akin, and A. Yilmaz, "Estimating driving behavior by a smartphone," in *Intelligent Vehicles Symposium (IV)*. IEEE, 2012, pp. 234–239.

[23] J. Dai, J. Teng, X. Bai, Z. Shen, and D. Xuan, "Mobile phone based drunk driving detection," in *4th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth)*. IEEE, 2010, pp. 1–8.

[24] M. Fazeen, B. Gozick, R. Dantu, M. Bhukhiya, and M. C. González, "Safe driving using mobile phones," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 3, pp. 1462–1468, 2012.

[25] P. Mohan, V. N. Padmanabhan, and R. Ramjee, "Nericell: rich monitoring of road and traffic conditions using mobile smartphones," in *Proceedings of the 6th ACM conference on Embedded Network Sensor Systems*. ACM, 2008, pp. 323–336.

[26] A. Ghose, P. Biswas, C. Bhaumik, M. Sharma, A. Pal, and A. Jha, "Road condition monitoring and alert application: Using in-vehicle smartphone as internet-connected sensor," in *10th International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 2012, pp. 489–491.

[27] A. Mednis, G. Strazdins, R. Zviedris, G. Kanonirs, and L. Selavo, "Real time pothole detection using android smartphones with accelerometers," in *Distributed Computing in Sensor Systems and Workshops (DCOSS), 2011 International Conference on*. IEEE, 2011, pp. 1–6.

[28] Y. Yang, B. Chen, L. Su, and D. Qin, "Research and development of hybrid electric vehicles can-bus data monitor and diagnostic system through obd-ii and android-based smartphones," *Advances in Mechanical Engineering*, vol. 2013, 2013.

[29] J. Zaldivar, C. T. Calafate, J. C. Cano, and P. Manzoni, "Providing accident detection in vehicular networks through OBD-II devices and Android-based smartphones," in *36th Conference on Local Computer Networks (LCN)*. IEEE, 2011, pp. 813–819.

[30] J. White, C. Thompson, H. Turner, B. Dougherty, and D. C. Schmidt, "WreckWatch: automatic traffic accident detection and notification with smartphones," *Mobile Networks and Applications*, vol. 16, no. 3, pp. 285–303, 2011.

[31] C. Thompson, J. White, B. Dougherty, A. Albright, and D. C. Schmidt, "Using smartphones and wireless mobile sensor networks to detect car accidents and provide situational awareness to emergency responders," in *ICST Conference, June*, 2010.

[32] V. C. Magaña and M. M. Organero, "Artemisa: Using an Android device as an eco-driving assistant," *Cyber Journals: Multidisciplinary Journals in Science and Technology: Journal of Selected Areas in Mechatronics (JMTC)*, 2011.

[33] R. Araujo, A. Igreja, R. de Castro, and R. Araujo, "Driving coach: A smartphone application to evaluate driving efficient patterns," in *Intelligent Vehicles Symposium (IV)*. IEEE, 2012, pp. 1005–1010.

[34] J. Wahlstrom, I. Skog, and P. Händel, "Risk assessment of vehicle cornering events in gnss data driven insurance telematics," in *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, Oct 2014, pp. 3132–3137.

[35] W. Johan, I. Skog, and H. Peter, "Detection of dangerous cornering in gnss data driven insurance telematics."

[36] G. Castignani, T. Derrmann, R. Frank, and T. Engel, "Driver behavior profiling using smartphones: A low-cost platform for driver monitoring," *Intelligent Transportation Systems Magazine, IEEE*, vol. 7, no. 1, pp. 91–102, Spring 2015.

[37] P. Händel, J. Ohlsson, M. Ohlsson, I. Skog, and E. Nygren, "Smartphone-based measurement systems for road vehicle traffic monitoring and usage-based insurance," *IEEE Systems Journal*, vol. 8, no. 4, pp. 1238–1248, Dec 2014.

[38] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan, "The pothole patrol: using a mobile sensor network for road surface monitoring," in *Proceedings of the 6th international conference on Mobile systems, applications, and services*. ACM, 2008, pp. 29–39.

[39] J. Tulusan, T. Staake, and E. Fleisch, "Providing eco-driving feedback to corporate car drivers: What impact does a smartphone application have on their fuel efficiency?" in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. ACM, 2012, pp. 212–215.

[40] "Alcohol involvement in fatal crashes 2000 (march 2002), dot hs 809 419, nhtsa," http://www-nrd.nhtsa.dot.gov/pubs/, (Accessed on 12/15/2016).

[41] "Sensor types | android open source project," https://source.Android.com/devices/sensors/sensor-types.html, (Accessed on 11/17/2016).

[42] "Discovery insure driving challenge," https://www.discovery.co.za/portal/individual/insure-driving-challenge-campaign, (Visited on 09/23/2015).

[43] F. RAHMAN, H. KUBOTA, and K. SAKAMOTO, "Comparative study of traffic calming design process," *Proceedings of the Eastern Asia Society for Transportation Studies*, vol. 2007, pp. 361–361, 2007.

[44] S. S. Adlinge and A. Gupta, "Pavement deterioration and its causes," *International Journal of Innovative Research and Development|| ISSN 2278–0211*, vol. 2, no. 4, pp. 437–450, 2013.

[45] "Why gps eats so much battery power | itworld," http://goo.gl/i5JyHu, (Visited on 07/12/2015).

[46] T. W. Jones, L. Marzen, and A. Chappelka, "Horizontal accuracy assessment of global positioning system data from common smartphones," *Papers in Applied Geography*, vol. 1, no. 1, pp. 59–64, 2015.

[47] T. Witte and A. Wilson, "Accuracy of non-differential gps for the determination of speed over ground," *Journal of biomechanics*, vol. 37, no. 12, pp. 1891–1898, 2004.

[48] H. Han, J. Yu, H. Zhu, Y. Chen, J. Yang, Y. Zhu, G. Xue, and M. Li, "Senspeed: Sensing driving conditions to estimate vehicle speed in urban environments," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 2014, pp. 727–735.

[49] T. A. Dingus, V. L. Neale, S. G. Klauer, A. D. Petersen, and R. J. Carroll, "The development of a naturalistic data collection system to perform critical incident analysis: an investigation of safety and fatigue issues in long-haul trucking," *Accident Analysis & Prevention*, vol. 38, no. 6, pp. 1127–1136, 2006.

[50] A. Zeeman and M. J. Booysen, "Combining speed and acceleration to detect reckless driving in the informal public transport industry," 2013.

[51] "Accelerometers - stmicroelectronics," http://www.st.com/content/st_com/en/products/mems-and-sensors/accelerometers.html?querycriteria=productId=SC444, (Accessed on 11/26/2016).

[52] "Samsung galaxy s7 edge teardown," http://www.techinsights.com/teardown.com/samsung-galaxy-s7-teardown/, (Accessed on 08/20/2016).

[53] D. Petrinovic, "Causal cubic splines: Formulations, interpolation properties and implementations," *IEEE Transactions on Signal Processing*, vol. 56, no. 11, pp. 5442–5453, 2008.

[54] D. Wenzel, "Lecture notes on multirate signal processing," oct 2015, the author attended a course on Multi Rate Signal Processing at the Technical University of Munich.

[55] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[56] "Maths - quaternions - martin baker," http://www.euclideanspace.com/maths/algebra/realNormedAlgebra/quaternions/index.htm, (Accessed on 12/01/2016).

[57] S. O. H. Madgwick, A. J. L. Harrison, and R. Vaidyanathan, "Estimation of imu and marg orientation using a gradient descent algorithm," in *2011 IEEE International Conference on Rehabilitation Robotics*, June 2011, pp. 1–7.

[58] S. Boonmee and P. Tangamchit, "Portable reckless driving detection system," in *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2009. ECTI-CON 2009. 6th International Conference on*, vol. 01, May 2009, pp. 412–415.

[59] T. Imkamon, P. Saensom, P. Tangamchit, and P. Pongpaibool, "Detection of hazardous driving behavior using fuzzy logic," in *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2008. ECTI-CON 2008. 5th International Conference on*, vol. 2, May 2008, pp. 657–660.

[60] J. Castellanos, A. Susin, and F. Fruett, "Embedded sensor system and techniques to evaluate the comfort in public transportation," in *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, Oct 2011, pp. 1858–1863.

[61] J. Schietekat and M. Booysen, "Detection of reckless driving in the sub-saharan informal public transportation system using acceleration-sensing telematics," in *EUROCON, 2013 IEEE*, July 2013, pp. 597–601.

[62] F. Bruwer and M. J. Booysen, "Vehicle acceleration estimation using smartphone-based sensors," 2015.

[63] S. Hemminki, P. Nurmi, and S. Tarkoma, "Gravity and linear acceleration estimation on mobile devices," in *Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, ser. MOBIQUITOUS '14. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2014, pp. 50–59. [Online]. Available: http://dx.doi.org/10.4108/icst.mobiquitous.2014.258034

[64] Y. Zhao, *R and data mining: Examples and case studies*. Academic Press, 2012.

[65] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Intelligent Data Analysis*, vol. 11, no. 5, pp. 561–580, 2007.

[66] N. Kasabov, *Foundations of neural networks, fuzzy systems, and knowledge engineering*. Cambridge, Mass: MIT Press, 1996.

[67] L. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning?i," *Information Sciences*, vol. 8, no. 3, pp. 199 – 249, 1975. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0020025575900365

[68] A. Liaw and M. Wiener, "Classification and regression by randomforest," *R News*, vol. 2, no. 3, pp. 18–22, 2002. [Online]. Available: http://CRAN.R-project.org/doc/Rnews/

[69] E. Keogh and A. Mueen, "Curse of dimensionality," in *Encyclopedia of Machine Learning*. Springer, 2011, pp. 257–258.

[70] M. Kuhn, "Caret package," *Journal of Statistical Software*, vol. 28, no. 5, 2008.

[71] C. Schreiner, H. Zhang, C. Guerrero, K. Torkkola, and K. Zhang, "A semi-automatic data annotation tool for driving simulator data reduction," in *Driving Simulation Conference, North America*. Citeseer, 2007, p. 9.